Steve Ciarcia

Part 1: Display/Receiver

Build a Gray-Scale Video Digitizer

An imaging system with remarkable features for the price

Video technology has always interested me. One look at all the monitors, TVs, and displays around my house suggests that it goes deeper than interest. Freudian views aside, this is not the first time I have covered video technology in a Circuit Cellar project. In previous articles, I have described high- and low-resolution video display systems and even a low-cost digital camera. However, the one project I've always wanted to do has eluded me. Until now, I have had to hold off on the presentation of a cost-effective general-purpose high-performance gray-scale video-digitizing "frame grabber."

I'll explain all this later, but the key terms for the moment are "gray scale" and "frame grabber." Such terms usually indicate commercial units costing thousands of dollars.

While some video digitizers are designed as peripherals for specific computers, virtually all digitizers endent and involve significant trade-offs in performance to maintain low cost. Generally, their digitizing speed is significantly less than the rate necessary to capture a video image as it is transmitted in real time (1/30 to 1/60 second). Instead, they must repeatedly sample many sequential video frames. Digitizers like these—sequential field scanning digitizers—can deal only with stationary objects in front of a camera and can take as long as 30 seconds to scan and record an image. Such digitizers are useless if you are working with moving objects.

Another factor to consider is how a digitizer represents the intensity of each pixel. Most low-cost digitizers meet the minimum video display capabilities of their host computer and digitize each element only as black or white. Some allow a limited gray scale. Higher-performance digitizers offer 64 or more levels of gray scale as well as a high digitizing speed.

Fortunately, both static memory and video integrated circuit technology have progressed to the point where I can finally offer a project that attempts to meet the level of perfection I have outlined. For the next two months, I will describe a complete digital video system. You can use it independently as a video camera digitizer and display (to implement a video telephone, for example), or you can connect it to any personal computer for tasks like image processing, character recognition, and desktop publishing graphics.

ImageWise consists of two separate boards: a digitizer/transmitter and a display/receiver. Each board can be used independently, or they can be connected to form a complete digitizer/transmitter/receiver system (see photo 1).

In contrast to other digitizers, ImageWise is a true frame grabber that takes only 1/60 second to capture an image. It accepts the video signals from devices like a standard TV camera (either monochrome or color), VCR, laserdisc player, and camcorder, and it then stores the picture as 244 lines of 256 pixels with 64 levels of gray scale—256 by 244 by 6 bits (see photo 2). The ImageWise digitizer/transmitter board converts the stored video image to RS-232 serial data that can be transmitted to any computer or to the ImageWise display/receiver board. Transmission rates are selectable from 300 bits per second up to 57.6k bps.

The ImageWise display/receiver board has a serial RS-232 input and a composite video output. It receives serial data directly from the digitizer/transmitter board or transmitted from a file downloaded from your computer and converts this data back into a picture on a composite video input black-and-white monitor (adding a pair of modems lets you send the images over telephone lines). The displayed image is an interlaced 256 by 244 by 6-bit gray-scale picture. The following specifications for the display/receiver board sum it up better:

- Resolution: The three selectable resolutions are 256 by 244, 128 by 122, and 64 by 61. All resolutions support 64 levels of gray scale (each picture element is represented by 6 bits). Note that, regardless of resolution, the system displays all pictures as interlaced full-screen images. Lower-resolution images are composed of larger pixel blocks.
- Video output: 75-ohm, 1.5-volt peak-to-peak composite video.
- Serial input: RS-232, 8 bits, 1 stop bit, no parity. Transmission rate is selectable from 300 bps to 28.8k bps.
- Hardware: 8031 microprocessor, Telmos 1852 video D/A converter, 64K bytes of static video RAM.

The specifications of the digitizer/transmitter board are

- Resolution: Same as above.
- Video input: 1-V peak-to-peak, black and white or color, 75-ohm termination.
- Serial output: RS-232, 1 start bit, no parity. Transmission rate is selectable from 300 bps to 57.6k bps.
- Hardware: 8031 microprocessor, RCA CA3306 6-bit flash A/D converter, 64K bytes of static RAM.

continued

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. The author of several books on electronics, he can be reached at P.O. Box 582, Glastonbury, CT 06033.

by the electron beam and continues to glow even after the beam passes on. Because the entire screen is scanned rapidly enough to get the beam back to each spot before the phosphor glow fades out, the entire screen seems to be illuminated at once.

One key difference between a TV display and most computer CRT displays is that the electron beam in a TV set can take on a wide variety of intensities, ranging from completely off (black) through shades of gray to completely on (white). A computer display may allow only black, one shade of gray, and white. We'll see what this difference means a little later on. Most composite video input amber or green monitors also have some gray-scale (or should I say green-scale?) capability.

As you might expect, the actual details are a bit more involved. A TV screen is scanned twice for each image, with the two sets of scan lines interlaced on the screen (computer displays are generally noninterlaced). This allows the whole screen to be scanned in half the time a noninterlaced scan would take, without reducing the number of lines in the image. Figure 2 shows how interlaced scanning paints lines on the screen. Each vertical scan is called a *field*, and two matched fields make up a *frame*. (1/50)

One field is completed in 1/60 second. (312.5) There are 262.5 lines in each field (see figure 2 to locate the half lines), so each line must be scanned in 63.5 microseconds (1/60 second divided by 262.5). (1/50 / 312.5) Figure 3 illustrates the signal voltage sent to the display for a single scan line, along with the allowable times for each part of the waveform. (Note: By strict definition, ImageWise is a field rather than frame grabber, since it digitizes only the first 244-line field of an interlaced frame. However, since the term frame grabber has come to mean a digitizer with the speed to digitize within the time period of a video frame, I shall continue to use the term frame grabber.)

The horizontal sync pulse occurs every 63.5 μs and tells the monitor to end the current line, return to the left edge, and begin another line. Surrounding each sync pulse is a blanking voltage that ensures that the track of the retrace will not be displayed on the screen. After allowing time for retrace and blanking, about 52 μs are left for the actual video picture on the horizontal scan line.

A vertical sync signal tells the monitor when to end the current field, return to the top of the screen, and begin the next field. Because the two fields in each frame are offset by exactly half a line, the timings at the end of each field are slightly different. Figure 4 diagrams the analog voltage required to display a complete field. To keep the size of the diagram down, only the first and last lines of active video are shown. The horizontal scale is distorted so that you can see the details.

You should note that the horizontal

continued

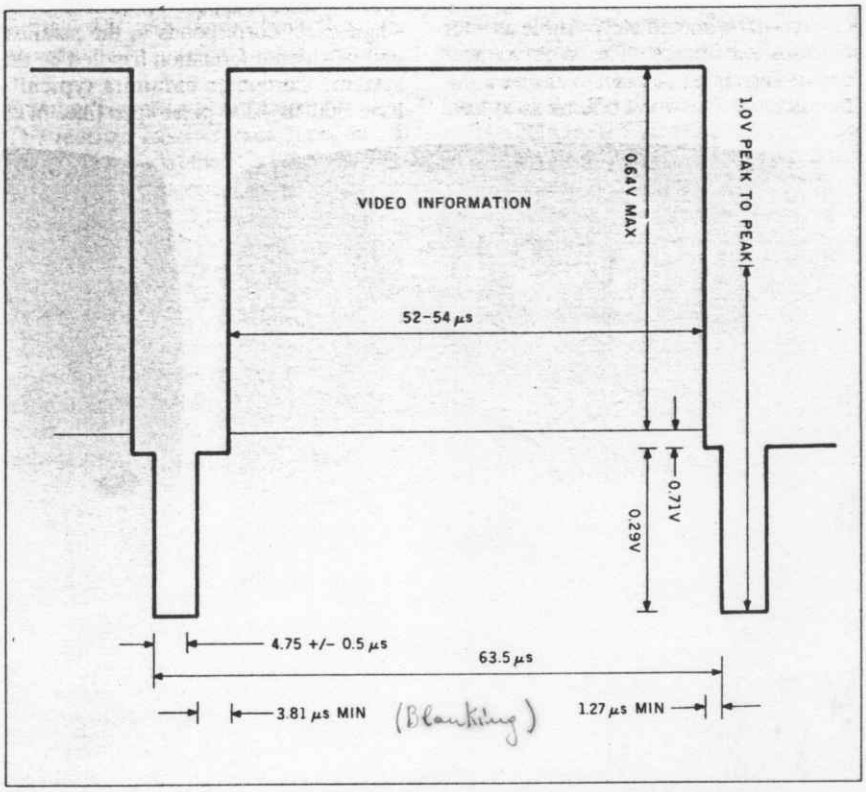


Figure 3: *A profile of the signal voltage sent to the display during a single scan line. Note the bracketing 63.5-microsecond horizontal sync pulses.*

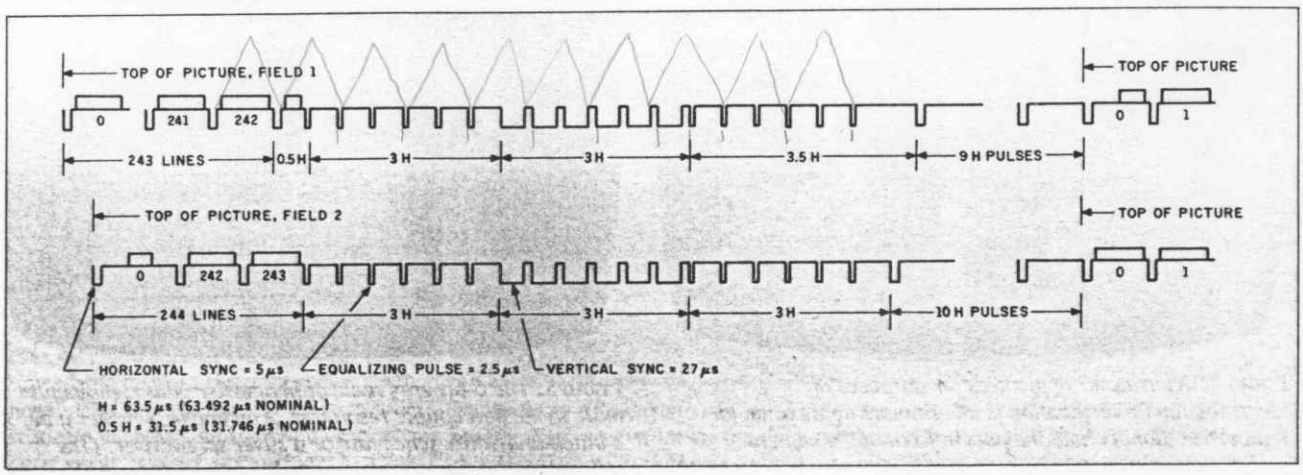


Figure 4: *Signal voltage timing diagram for a complete field.*

sync pulses do not stop during the vertical sync and retrace. In fact, to make sure that the monitor switches smoothly across the half line, sync pulses occur every 31.75 μs during most of the vertical sync period.

A blanking voltage surrounds the vertical sync pulse itself to ensure that the vertical retrace is not visible on the screen. Each field contains 244.5 visible lines, which you can verify by examining the timing diagram in figure 4.

Although video monitors can tolerate small variations in the number of displayed lines or the exact line timings, any errors will be immediately visible as jitter or distorted images. The sync voltages and timings must be exact to ensure a stable picture. The worst offense is to have timings that vary "just a little" from field to field; this will cause the picture to jitter annoyingly.

Variations on a Theme

Now that you're acquainted with standard video, I can explain some of the basic design criteria for ImageWise. As with any project, there are trade-offs between "the ultimate system" and "the one that got built." I will try to explain why I made the decisions I did.

The most basic question was one of resolution: How many picture elements (pixels) should appear on each line? A single pixel corresponds to the smallest unit of video information handled by the system. Computer monitors typically have 300 to 1000 pixels per line. With about 50 μs available in each line to display those pixels, a 1000-pixel line requires a new pixel every 50 nanoseconds. Since typical dynamic RAMs have a cycle time of 300 ns, allowing a few nanoseconds for the other circuitry would require about eight banks of RAM to ensure that a pixel was ready every 50 ns. Using 50-ns static RAMs, while feasible, would be very expensive.

I decided to look at it the other way: How many pixels will fit on a line with affordable hardware? We find that 32K-byte static RAMs are increasingly affordable, and they are considerably faster than DRAMs. Since even the "slow" ones have a cycle time of about 130 ns, one pixel every 200 ns is reasonable. That allows 256 pixels in 51.2 μs. Be-

Photo 2: *The ImageWise frame-grabber (or "freeze-frame digitizer") captures a video signal in 1/60 second; fast enough to digitize live TV broadcast signals as easily as those from a stationary camera.*

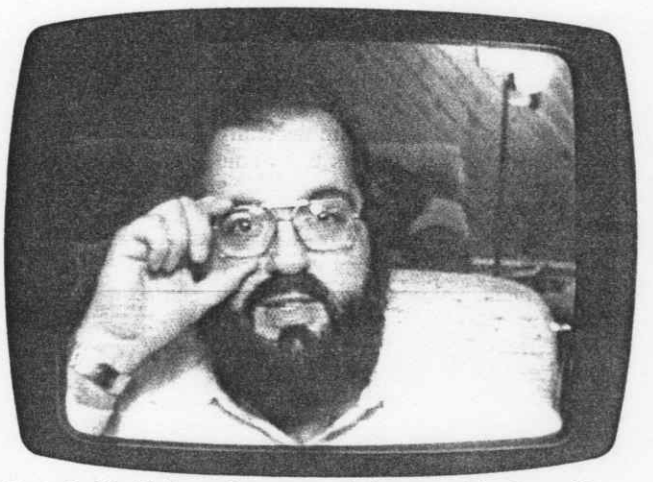


Photo 3: *The high-quality gray-scale images of the ImageWise digitizer can be used in security, pattern recognition, a video telephone, and image database applications. (Phone me sometime; I may be looking back at you through the camera.)*

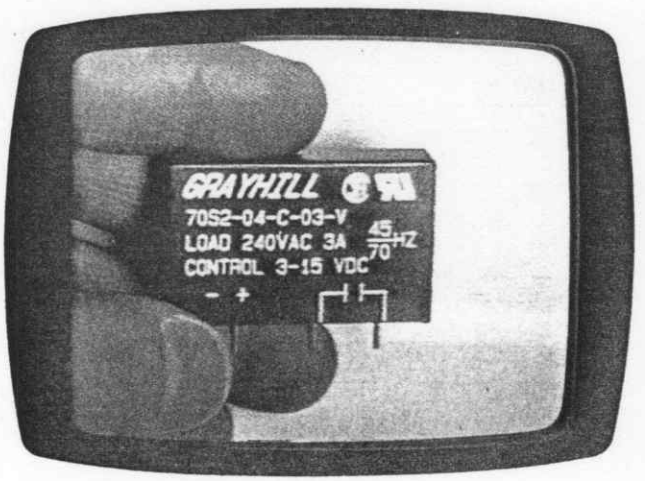


Photo 4: *Teletransfer of pictures for purposes of identification or verification is a legitimate application for ImageWise. Simply hold the part in front of the camera and transmit the picture to everyone.*

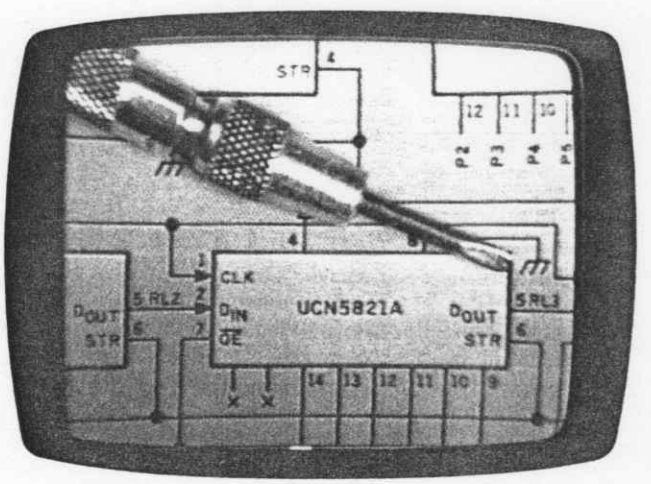


Photo 5: *The 6-bit gray scale of ImageWise adds significantly more to the perceptible resolution of an image, whether it be a black-and-white schematic or a silver screwdriver. (The schematic and the screwdriver could not be represented accurately without gray scale.)*

cause 256 is a "magic" number, I knew I was on the right track.

As I mentioned earlier, each field has about 244 visible lines. Therefore, a 64K-byte buffer could hold one field with some room left over. Two fields could be contained in 128K bytes. With two fields, however, the vertical resolution (488) is twice the horizontal resolution (256). This seemed excessive. Fortunately, because both fields often contain redundant information, I decided to keep the amount of RAM within reasonable bounds and digitize only a single field. But how would a 256- by 244-pixel picture look compared to the original?

All my experience with 320 by 200 computer displays suggested that I might not like the results and be forced to go back to expensive plan A. However, seeing is believing, so I figured I'd build it and decide then. (Often it is easier and faster to build a prototype and take a look at the results than to argue about what might be.)

Photo 3 shows the quality of the image I got with a 256 by 244 display. (So much for my prior experience with computer displays!) There's a good reason why I was wrong, and if you're as surprised as I was, here's the explanation.

You see jagged diagonal lines or "jaggies" on low-resolution computer displays because each pixel can have only a few levels of brightness. The jaggies can be reduced only by increasing the number of pixels on each line. Depending upon the subject material, resolutions of 640 pixels per line and 350 to 400 lines per screen are required to see noticeable improvement.

But there is another way to reduce the jaggies: If each pixel can take on many levels of brightness, the sharpness of the edges can be reduced. ImageWise uses 6 bits to represent each pixel, allowing 64 shades of gray. Real-world images don't have crisp, computer-generated edges, so each pixel tends to shade into the adjoining ones. The effect is a rather smooth picture that has more "effective" resolution than you'd expect (see photos 4, 5, and 6). This is why pictures shown on color displays that incorporate palette D/A converters often look better. Look closely at a line boundary and see if there is some gradual shading.

As an example, you're probably aware that a standard TV does not make a good computer monitor. Trying to fit more than 40 characters on a line results in an unreadable display. However, newspaper headlines displayed on a TV are easily readable even though the characters are very small, simply because each pixel can take on many brightness levels. Watch your TV carefully and see.

Finally, why does ImageWise use 6 bits per pixel and not more if gray scale is such a good idea? Again, it is a cost tradeoff. We have to digitize and determine the gray-scale value of 256 data points in 50 μs, or one every 200 ns. This requires a fast A/D converter called a flash A/D converter. The price of one is directly related to the number of bits it resolves. Eight-bit models are considerably more expensive than 4- or 6-bit chips. The device I ultimately chose was the RCA 3306 6-bit flash A/D converter, which can operate at 12 million to 16 million samples per second (our sample rate is 5 megahertz). I'll talk more about this next month.

Display/Receiver Hardware

The receiver has two main functions: It accepts data from the RS-232 serial port and displays the resulting picture on a monitor. Figure 5 shows the receiver hardware.

As in many recent Circuit Cellar projects, the receiver uses an Intel 8031 single-chip microprocessor to control the rest of the hardware. A 2764 EPROM stores the 8031's program. An Intel 8254 Programmable Interval Timer (PIT) produces the sync pulses. The video field data is held in a pair of 32K-byte static RAMs and is converted to an analog voltage by a specialized video D/A converter. The MC145406 converts RS-232 voltages into TTL levels for the 8031's serial port.

It was tempting to use the 8031 to produce the sync pulses directly, but a little study showed that there was no way to get the precise timings required for a stable picture. The 8254 is connected to produce repetitive pulses, so the 8031 need only program the appropriate values into the 8254's registers when a change is required. The 8254 uses a 500-ns (2-MHz) clock divided from the 10-MHz crystal oscillator. Figure 6 shows the 8254 pulses for a normal video line.

Often it is easier and faster to build a prototype and take a look at the results than to argue about what might be.

The Telmos 1852 is a specialized video D/A converter that accepts up to 8 video data bits, a blanking input, and a sync input. The analog output conforms to the standard video specifications. Using this D/A converter eliminates a lot of hardware that would otherwise be required to combine the video, blanking, and sync signals to produce the right output voltage with enough power to drive the monitor. The 8031 ensures that the 2 unused bits (the low-order ones) are always 0.

The 16-bit address required by the 64K-byte field buffer is divided into two parts: a high byte supplied by the 8031 and a low byte that can come from either the 8031 or an 8-bit counter. Normally, the counter steps through the 256 pixels on each line, and the 8031 counts out the lines in the high byte. Both bytes are supplied by the 8031 when it reads or writes buffer data.

A pair of LS244s isolate the field buffer's data bus from the 8031's data bus,

continued

Photo 6: *An in/out comparison. The camera is pointed at the magazine in the middle. The monitor on the left displays the image seen by the camera and the digitizer/transmitter. The monitor on the right shows the digitized image received by the display/receiver board.*

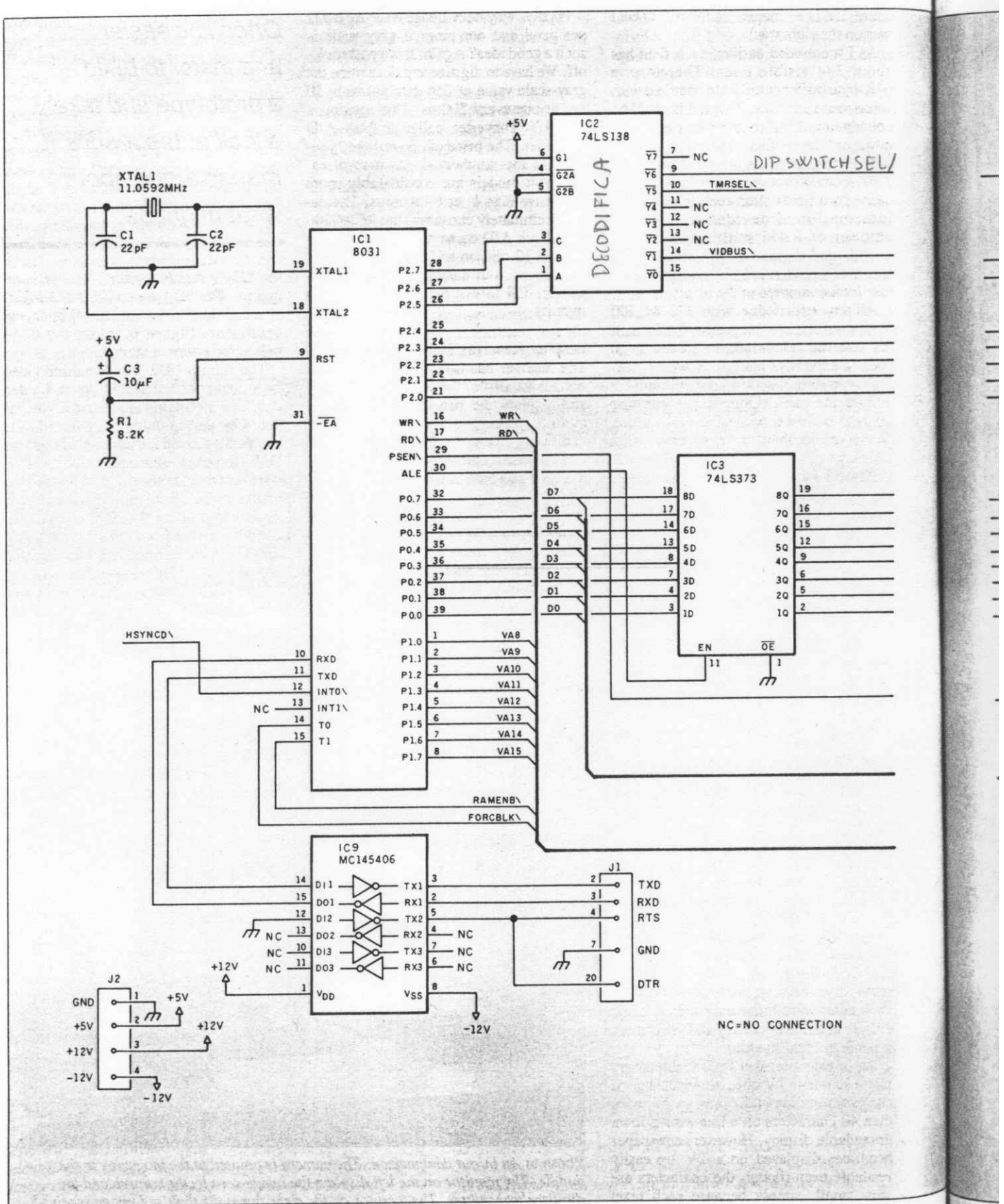
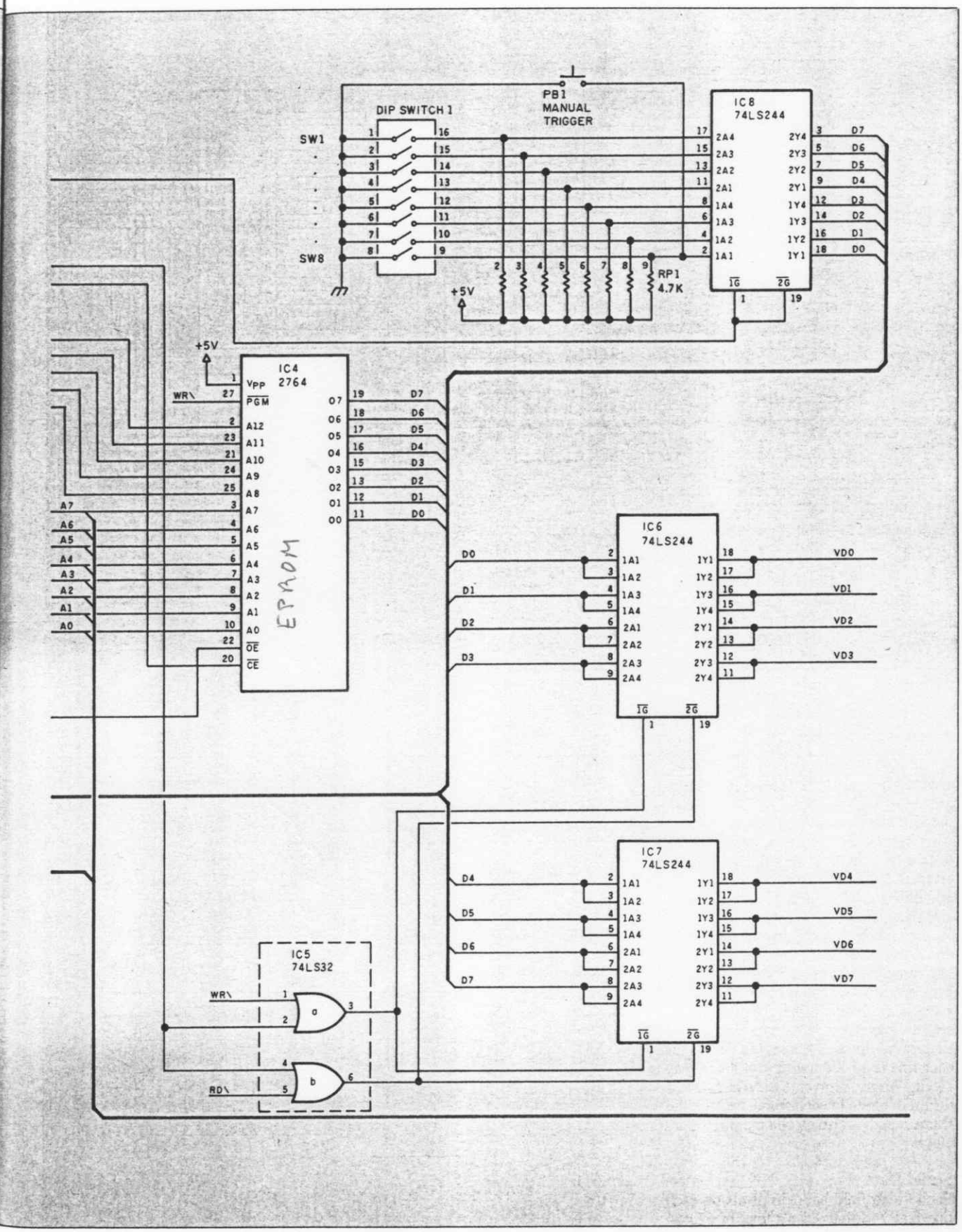


Figure 5: *Schematic diagram of the ImageWise display/receiver hardware.*


DIP SWITCH 1
PB1
MANUAL
TRIGGER
SW1
SW8
RP1
4.7K
+5V
IC8
74LS244
IC4
2764
EPROM
IC6
74LS244
IC7
74LS244
IC5
74LS32
WR\
RD\
D0
D1
D2
D3
D4
D5
D6
D7
VD0
VD1
VD2
VD3
VD4
VD5
VD6
VD7

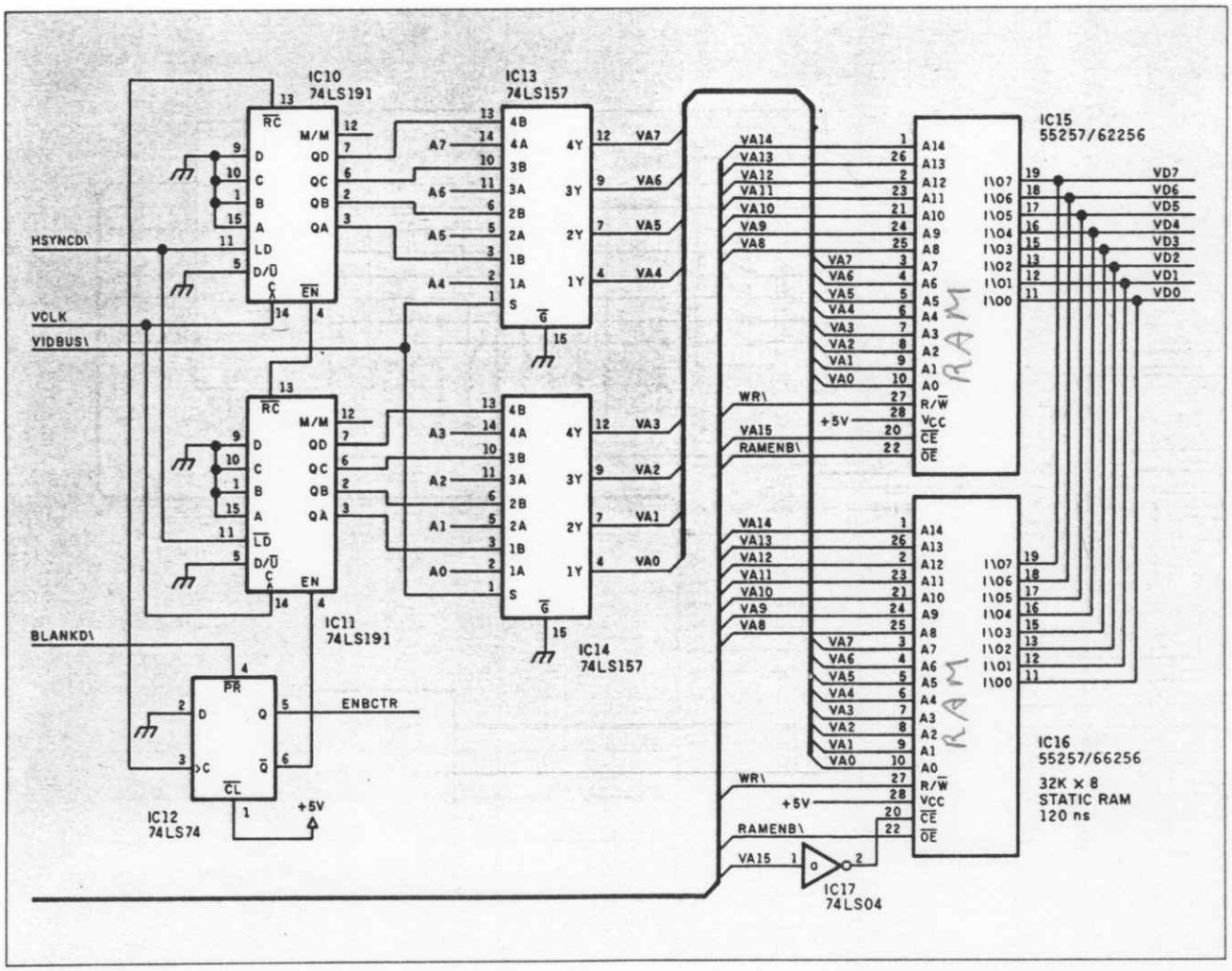


Figure 5: *Continued.*

except when the 8031 is reading or writing to the buffer. A third LS244 connects the DIP switches to the 8031's data bus. An 11.059-MHz crystal allows the 8031 to receive and generate standard RS-232 bit rates. The video data and sync timings are derived from a separate 10-MHz crystal oscillator circuit.

The divide-by-two counter that produces the video data clock is reset by the horizontal sync pulses from the 8254. This ensures that the pixel clock has the same phase in each line. Without the reset, the clock would alternate phases in successive lines because the length of each line is an odd multiple of the 500-ns clock. Worse, a given line would have a different phase in each frame because the frame length is also an odd multiple of the clock.

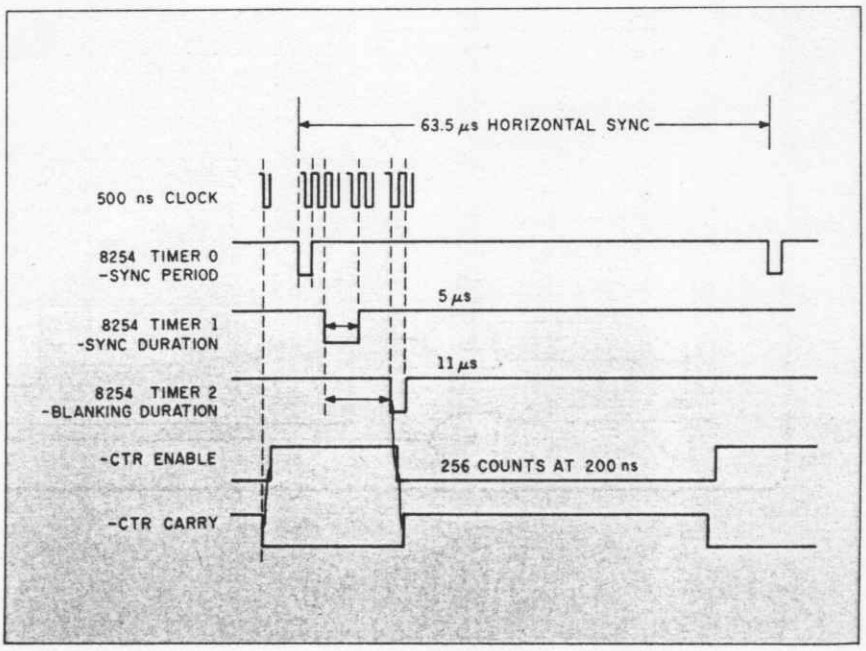


Figure 6: *Timing pulses generated by the display/receiver's 8254 PIT.*

Serial Data

Each video field has 256 pixels on each line and 244 lines, for a total of 62,464 pixels (we round the half line up to a full

continued

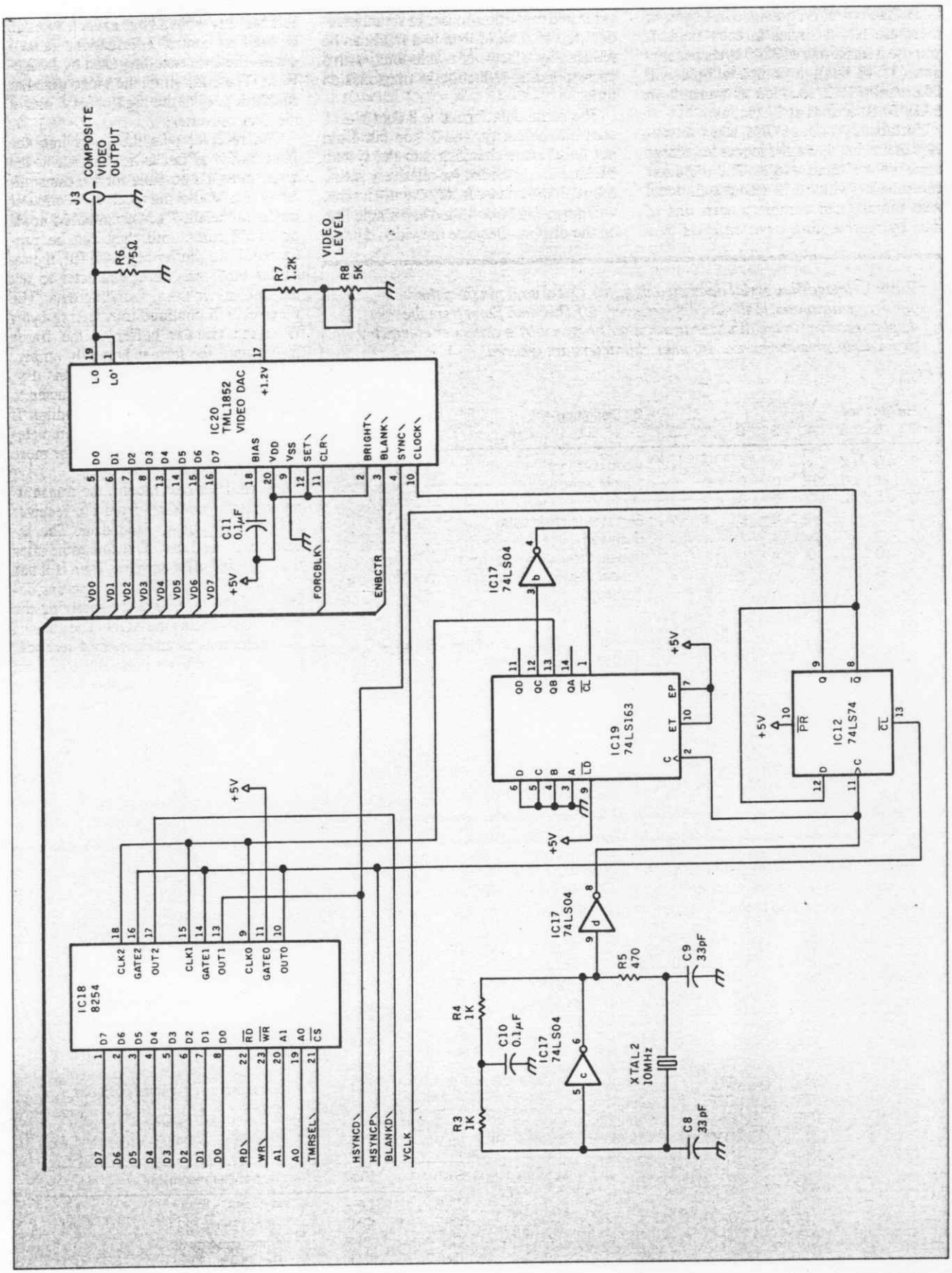


Figure 5: *Continued.*

line). Each pixel is contained in 1 byte, so there are 62,464 bytes in each field. If you use a serial rate of 3840 bytes per second (38.4k bps), a complete field will take about 16.2 seconds to transmit (it takes 10.8 seconds at 57.6k bps).

Fortunately, ImageWise takes advantage of the fact that most scenes have large areas of the same shade. The digitizer/transmitter (which I'll describe in detail next month) can compress each line of data by representing repeated bytes by a value and repetition count. In actual practice, the amount of data in a field can be reduced by a factor of two to four, with a corresponding reduction in transmission time.

The serial data format is 8 data bits, 1 start bit, no parity, and 1 stop bit. I did not build error checking into the system because it is intended for relatively short, robust connections. In any event, an error will generally be confined to a single line on the display. Because the video data itself has only 6 bits, 2 bits in each byte can be used for control information. Table 1 details the byte encoding used by ImageWise. The 8031 shifts the video data left so that it goes to the high-order 6 bits of the D/A converter.

The receiver puts the bytes into the field buffer as fast as it can, but at the faster rates it's possible for the transmitter to get ahead of the receiver. A circular buffer in the 8031's internal RAM holds up to 48 bytes until they can be processed. If this buffer begins to fill, the receiver sends an XOFF character to tell the transmitter to stop sending data. The receiver will continue transferring bytes from the circular buffer to the frame buffer until the former is nearly empty. Just before the circular buffer runs dry, the receiver sends an XON character to tell the transmitter to resume sending. If the circular buffer does empty completely, the receiver will simply wait for more bytes to show up.

As we'll see next month, the transmitter waits for an XON from the receiver before beginning to send data. The receiver will send the XON sometime after the circular buffer empties, even if it has sent one before. A DIP-switch setting determines the time between emptying the buffer and sending the XON. The choices are continuous pictures, every 4 seconds, every 8 seconds, or manually triggered (see table 2). A push button is used to trigger a new field from the transmitter in manual mode.

Table 1: *ImageWise serial data encoding. (a) This data flows from the digitizer/transmitter to the display/receiver. (b) This data flows from the display/receiver to the digitizer/transmitter or is sent by a computer connected to the digitizer/transmitter. All other characters are ignored.*

(a)

Bit number 7	6	5	4	3	2	1	0	Bit Definition
0	0	x	x	x	x	x	x	Video data byte
0	1	0	0	0	0	0	0	Start of video field
0	1	0	0	0	0	0	1	Start of video line
0	1	0	0	0	0	1	0	End of video field data
0	1	1	x	x	x	x	x	Reserved
1	0	0	0	x	x	x	x	Repeat previous byte *x* times (0 = 16 reps)
1	0	0	1	x	x	x	x	Repeat previous byte 16*x* times (0 = 256 reps)
1	1	x	x	x	x	x	x	Reserved

(b)

Bit number 7	6	5	4	3	2	1	0	Bit Definition
0	0	0	1	0	0	0	1	XON, starts or restarts transmission
0	0	0	1	0	0	1	1	XOFF, halts transmission
1	0	0	0	0	0	0	0	Use 256 by 244 resolution (full)
1	0	0	0	0	0	0	1	Use 128 by 122 resolution (half)
1	0	0	0	0	0	1	0	Use 64 by 61 resolution (quarter)

The Software

It's worthwhile to describe how the software pulls the receiver hardware together. The two main jobs are maintain-

Table 2: *Receiver DIP-switch settings. ON and OFF refer to switch positions. (a) SW1, SW2, and SW3 select the serial bit rate (must match transmitter rate). (b) SW4 and SW5 select the time-out interval. (c) SW6 and SW7 select the transmitter resolution. (Note: A manual push button is connected to the SW8 position, so SW8 must be OFF.)*

(a)

SW1	SW2	SW3	Serial transmission rate (bits/second)
OFF	OFF	OFF	300
OFF	OFF	ON	600
OFF	ON	OFF	1200
OFF	ON	ON	2400
ON	OFF	OFF	9600
ON	OFF	ON	19.2k
ON	ON	OFF	28.8k
ON	ON	ON	57.6k

(Note: 4800 bps intentionally omitted.)

(b)

SW4	SW5	Time-out interval
OFF	OFF	Continuous pictures, no delay
OFF	ON	4-second delay between pictures
ON	OFF	8-second delay between pictures
ON	ON	Send picture by manual push-button trigger

(c)

SW6	SW7	Transmitter resolution
OFF	OFF	Full: 256 by 244
OFF	ON	Half: 128 by 122
ON	OFF	Quarter: 64 by 61
ON	ON	Reserved

ing stable video sync and accepting bytes from the serial interface. The code is written in assembly language to maximize the performance of the 8031. Figure 7 is a flowchart of the software's major components.

The 8254 PIT generates the precise sync signals for each line, so the 8031 need only reprogram the PIT when a change is needed. Because changes to the 8254's settings take effect with the next 8254 sync output, the 8031 must make the changes one sync pulse before they're actually needed. All timings are determined by counting sync pulses, which are connected to the 8031's INT0 interrupt request pin.

The INT0 interrupt handler decrements a counter and checks to see if it's 0. If so, an 8254 change is required; otherwise, the handler simply returns to the mainline code. Each change to the 8254 involves writing a few bytes and reloading the counter to tell how many interrupts will pass before the next change.

Each 8031 instruction takes 1 or 2 μs. At most, only about 50 instructions can be executed per horizontal line. During the vertical retrace interval the sync pulses are only 31.5 μs apart, giving time for only 20 instructions per sync pulse. The interrupt routine must have enough

continued

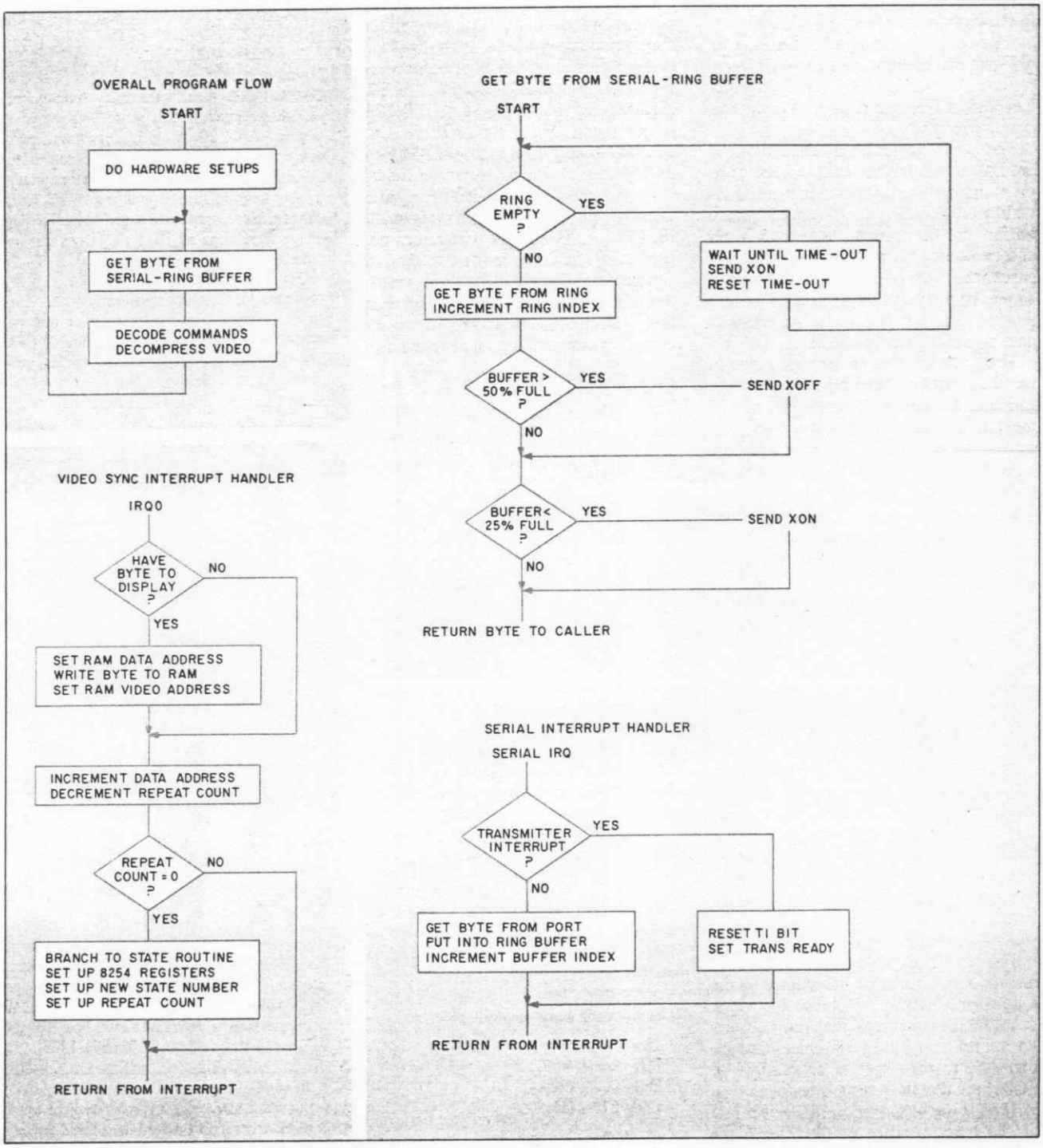


Figure 7: *Flowchart for the ImageWise display/receiver system's software.*

time to get ready for the 8254 loading during the short sync pulses in the vertical retrace interval, so control is passed to the routine two sync pulses before the change is needed. The interrupt routine then uses a polling loop to detect the last sync pulse.

Another interrupt is generated within the 8031 whenever a byte is received on the serial port. This interrupt awakens the serial interrupt handler routine, which reads the byte from the port and places it in the circular buffer in the 8031's internal RAM. The serial interrupt handler has a lower priority than the sync interrupt; consequently, the serial interrupt handler can be interrupted whenever a sync pulse occurs.

The sync and serial interrupt routines are linked by a background task that simply waits for bytes to show up in the circular buffer. Whenever a byte appears in the buffer, the background task takes it out and decides what to do with it. In most cases, the byte is either video data that should be put in the field buffer or a count that tells how many times the previous data byte should be repeated.

The ordinary way of putting a byte in the field buffer would be to have a subroutine that saves all the registers, sets up the buffer address, does the write, restores the registers, and returns to the caller. Unfortunately, this scheme doesn't work in our application because the writes to the frame buffer have to occur just after the video syncs to reduce sparkles in the display. Additionally, the sync interrupt routine must get control at the same time to reset the 8254. Something has to give!

The solution is to combine the two functions in the video sync interrupt handler. Whenever the background routine has a byte to be written in the buffer, it sets up the registers and turns on a flag bit. The sync routine checks the flag, does the write if it's on, then turns the flag off. The background routine sits in a loop until it sees that the flag has been reset, then continues on its way. Because the background routine has handled all the register setups, the interrupt routine can proceed at full speed and write the byte immediately without saving or restoring any registers.

The possibility arises that the serial interrupt handler will be interrupted by the video sync handler. Because the video sync handler assumes that the registers are set up for it, the serial interrupt handler has to take special precautions to make sure that the wrong byte doesn't get written at the wrong address.

The sync interrupt handler checks the switches once every frame (at the end of the second field) to see if anything's changed. If so, it drops what it's doing and runs through the power-on initialization routine again. If a picture is being received when you flip the switches, it will get garbled because the serial port will miss a few characters. The rule of thumb is to change switch settings only when nothing else is happening.

Experimenters

While printed circuit boards and kits are available for the ImageWise system, I encourage you to build your own. If you don't mind doing a little work, I will support your efforts as usual. A hexadecimal file of the executable code for the 8031 digitizer and display system EPROMs, sample picture files, and the Turbo Pascal code for storing images on an IBM PC or SB180 are available for downloading from my BBS at (203) 871-1988. Alternatively, you can send me a preformatted IBM PC or SB180 disk with return postage, and I'll put the files on it for you (the hexadecimal file could be used with my serial EPROM programmer, for example). Of course, this free software is limited to noncommercial personal use.

Circuit Cellar Feedback

This month's feedback begins on page 58.

Next Month

Having a gray-scale video display is one thing, but where do you get all the pictures? I'll describe the digitizer/transmitter hardware that captures images from a camera or TV and sends them to either the receiver for immediate display or a computer for storage. ■

Special thanks to Ed Nisley for his expert collaboration on this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, P.O. Box 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

CCI
P.O. Box 428
Tolland, CT 06084
(203) 875-2751

1. ImageWise digitizer/transmitter board experimenter's kit. Contains digitizer/transmitter printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with transmitter software, and CA3306 flash A/D converter and manual with complete parts list.
DT01-EXP $99

2. ImageWise display/receiver board experimenter's kit. Contains gray-scale display/receiver printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with receiver software, Telmos 1852 video D/A converter, manual with complete parts list, and an IBM PC 2.0 disk containing sample digitized images and test patterns.
DR01-EXP $99
DT01-EXP and DR01-EXP together $179

3. ImageWise digitizer/transmitter full kit. Contains all digitizer/transmitter components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, CA3306 flash A/D converter, manual, and IBM PC 2.0 disk containing utility routines for storing and displaying (dot-dithered, not gray scale) and downloading image files using an IBM PC. Does not include power supply or case.
DT01-KIT $249

4. ImageWise display/receiver full kit. Contains all gray-scale display/receiver components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, Telmos 1852 video D/A converter, manual, and an IBM PC 2.0 disk containing sample digitized images and test patterns. Does not include case or power supply.
DR01-KIT $249
DT01-KIT and DR01-KIT together $489

While only kits are described above, ImageWise has been licensed for assembly. Call CCI for source and availability of assembled boards and complete systems, black-and-white TV cameras, 32K-byte static RAM chips, and power supplies. Software utilities are also available in SB180 format.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Surface delivery (U.S. and Canada only): add $3 for U.S., $6 for Canada. For delivery to Europe via U.S. airmail, add $10. Three-day air freight delivery: add $8 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express) . Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

To be included on the Circuit Cellar mailing list and receive periodic project updates and support materials, please circle 100 on the Reader Service inquiry card at the back of the magazine.

Steve Ciarcia

Part 2: Digitizer/Transmitter

Build a Gray-Scale Video Digitizer

An imaging system with remarkable features for the price

Last month I described the ImageWise video digitizer's display/receiver section. The display/receiver board accepts binary data from a serial RS-232 port and decodes that data to generate a gray-level display (with 64 levels) on a standard TV monitor. This month I'll complete the project by describing the digitizer/transmitter board (see photo 1) and discussing some possible applications. (Note: Certain portions of this article depend heavily on material presented in last month's Circuit Cellar.)

As I mentioned last month, ImageWise is technically a "field" grabber rather than a "frame" grabber. The digitizer/transmitter board makes no distinction between the two fields in a frame: One is as good as the other. The digitizer/transmitter must decide when a new field is starting, wait until the first active line begins, then begin converting the analog video signal into digital bytes that are stored into the field buffer. Because the video can't be "slowed down," all this must happen when the video occurs rather than when the processor is ready.

You might think that you could locate the start of the first active video line by counting horizontal sync pulses after the conclusion of the vertical sync pulse, but it's not that easy. Some cameras do not produce "standard-width" vertical sync pulses, so counting pulses won't work. Instead, I used an internal timer on the 8031 to provide a fixed delay period (DIP-switch-selectable, either 16- or 20-line times). The first horizontal sync pulse after that delay becomes the first active line to be digitized.

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. The author of several books on electronics, he can be reached at P.O. Box 582, Glastonbury, CT 06033.

Digitizer/Transmitter Hardware

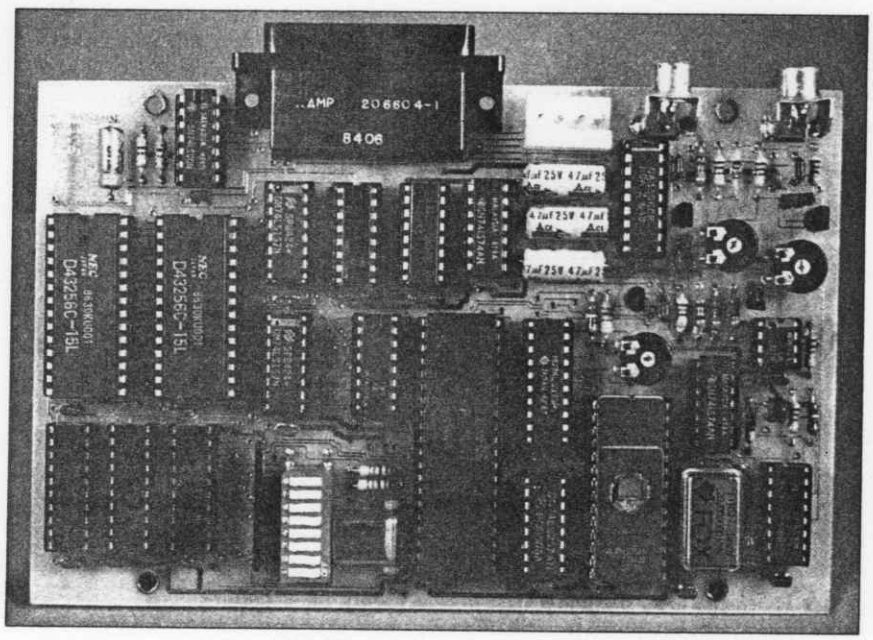
The digitizer/transmitter has two main functions. First, it digitizes the analog video signal; second, it transmits the data to the display/receiver over a serial RS-232 link. Figure 1 shows the digitizer/transmitter circuitry.

The analog circuitry merits a detailed description. Many people who are familiar with the level of integration possible in digital circuitry are appalled at the number of components needed to accomplish even the simplest analog task. The whole mass of circuitry attached to the analog video input performs five functions: termination, clamping, filtering, buffering, and level comparison.

Standard composite video is communicated over coaxial cable with a character-

continued

Photo 1: *The ImageWise digitizer/transmitter in prototype printed circuit form. The digitizer/transmitter board flash-digitizes the video output of a TV camera or other video source (connectors in upper right corner) and converts it to serial data that can be stored and manipulated by a computer or redisplayed on an ImageWise display/receiver board (see last month's Circuit Cellar). The flash A/D converter is the 18-pin chip in the upper right corner; the two 28-pin chips on the left are 64K bytes of static RAM.*



istic impedance of 75 ohms. To prevent reflections from the end of the cable, it must be terminated in that impedance. The 75-ohm resistor at the video-input connector of the digitizer/transmitter (J3 and J4) accomplishes this goal. A jumper (JP1) disconnects the terminator if you have a terminated device (perhaps a monitor to watch "live" video) connected to the second parallel connector.

Video signals never seem to exhibit "textbook" profiles. Frequently, inexpensive cameras produce video signals with a DC offset that depends on the scene being viewed. The digitizer incorporates clamping circuitry that forces the tips of the sync pulses to 0 volts. This means that the brightest white will be in the area of 1 to 1.4 V (different cameras give different results).

Both color and black-and-white sig-

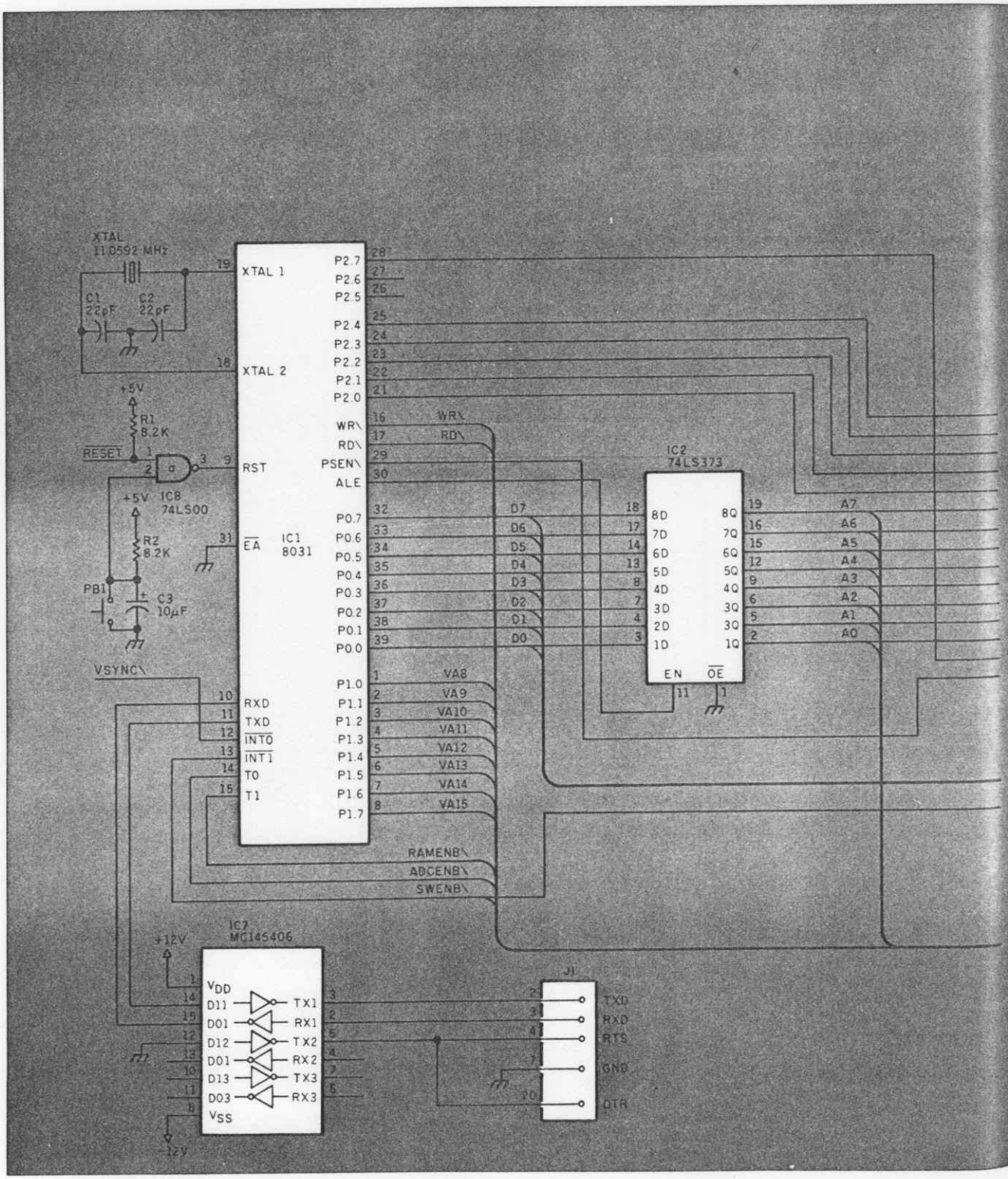
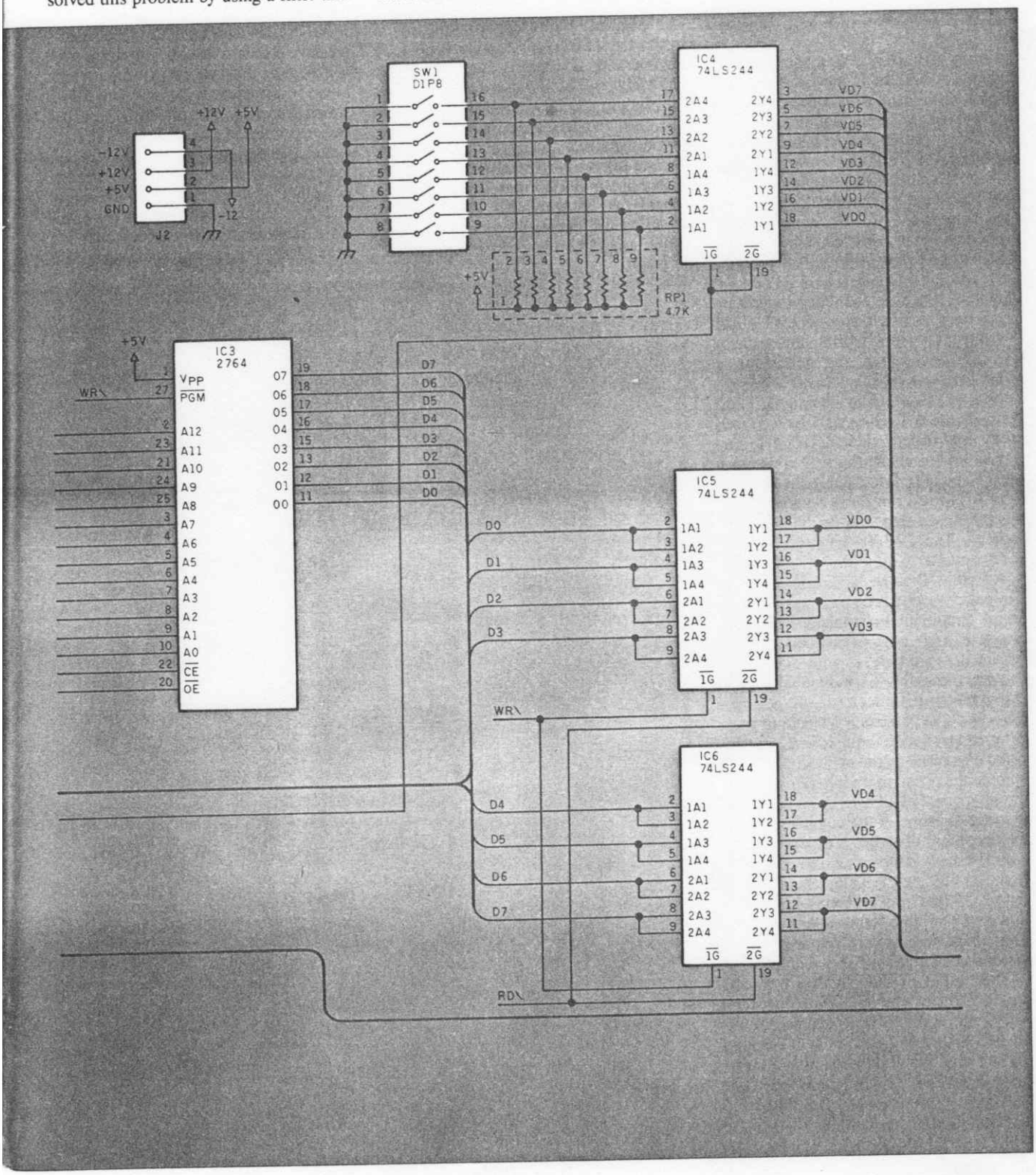


Figure 1: *Schematic for the ImageWise digitizer/transmitter board.*

nals will work with the ImageWise digitizer; however, a color signal contains more information than is necessary. This extra color information is detrimental because it can impart a herringbone pattern on the black-and-white digitized image. I solved this problem by using a filter that removes the frequencies used to encode the color. The remaining signal contains the important intensity voltage that the digitizer measures. You should disconnect this color filter (jumper-selectable JP2) when using a black-and-white camera because the filter doesn't have a sharp cutoff, and it can "soften" the picture by removing some of the finer details. Try it both ways and use the jumper setting that works best for your application.

The 2N4401 transistor serves as a power amplifier to ensure that the re-

continued



A flash A/D converter differs from slower converters in having more internal circuitry.

maining circuitry gets a clean signal without loading the input. Configured as an emitter follower, this transistor circuit supplies the relatively high input current required by the RCA CA3306 flash A/D converter.

A composite video signal contains both video and synchronization information. I used an LM311 comparator configured as a sync detector. Whenever the video drops below 200 millivolts (set by the resistors on pin 3), the LM311 output goes low. Because the clamping circuit forces the sync tips to ground, the LM311 output goes low only during horizontal or vertical sync pulses and never during video data.

Next, the buffered video signal is directed to the A/D converter. As we've already established, video data is extremely fast, and therefore the A/D conversion must be equally fast. I chose the CA3306 6-bit A/D converter for reasons of economy and the existence of readily available sources. The CA3306 is a special flash A/D converter.

A flash A/D converter is different from slower converters because it has more internal circuitry. Rather than use a D/A converter as an integral component in the conversion process, a 6-bit flash A/D converter contains 64 individual voltage comparators, each set to trigger at a specific level. Sophisticated decoding logic determines which comparator is triggered as a result of the applied signal and outputs the appropriate binary code. The ultimate speed of a flash A/D converter is the reaction time of the comparators and the decoding logic and is independent of any system clocks or other timing signals. In the case of the CA3306, its conversion time is 55 to 83 nanoseconds, or 12 million to 18 million samples per second. The ImageWise system requires a converter that samples at 5 megahertz.

The output of the CA3306 is 6 bits within a relative range defined between $+V^{ref}$ and $-V^{ref}$. Most often these limits are +5 V to +8 V and ground. Because we are digitizing only 64 gray levels, however, it is worthwhile to include only the active video range of +0.2 V to +1.5 V. This is easily accomplished with the white ($+V^{ref}$) and black ($-V^{ref}$) trim potentiometers that adjust the CA3306's conversion thresholds so that a bright white will be digitized as hexadecimal 3F and dark black will be hexadecimal 00. I'll describe the adjustment procedure later.

Finally, the delay trim potentiometer (R21) determines the blanking delay from the start of each horizontal sync pulse, which must be adjusted to match the camera. Video conversion begins when the LS221 one-shot times out. It ends exactly 256 pixels later. IC20 is a 20-MHz oscillator that is divided by four to produce the 200-ns clock that drives the pixel counters and supplies the RAM write signal.

The digital portion of the transmitter board is similar to that of the receiver board (see last month's article). Both boards must process the video data in the same way, albeit in opposite directions, so much of the circuitry can be the same.

Without the 8254 counter/timer used on the receiver board, fewer control lines were needed, so I was able to eliminate the 74LS138 decoder. Two 74LS244s isolate the processor data lines from the video data lines, and a third 74LS244 buffers the option switches. An 11.059-MHz crystal lets the 8031 receive and generate standard RS-232 bit rates.

As I mentioned earlier, the input video must be sampled at a rate of 5 MHz, which translates to around 200 ns between samples. A check of the 8031's execution speed shows there is no way that the processor can read the A/D converter, set up the proper address in memory, and store the byte in 200 ns. I chose instead to use the 8031 to select which line is currently being scanned and set up that

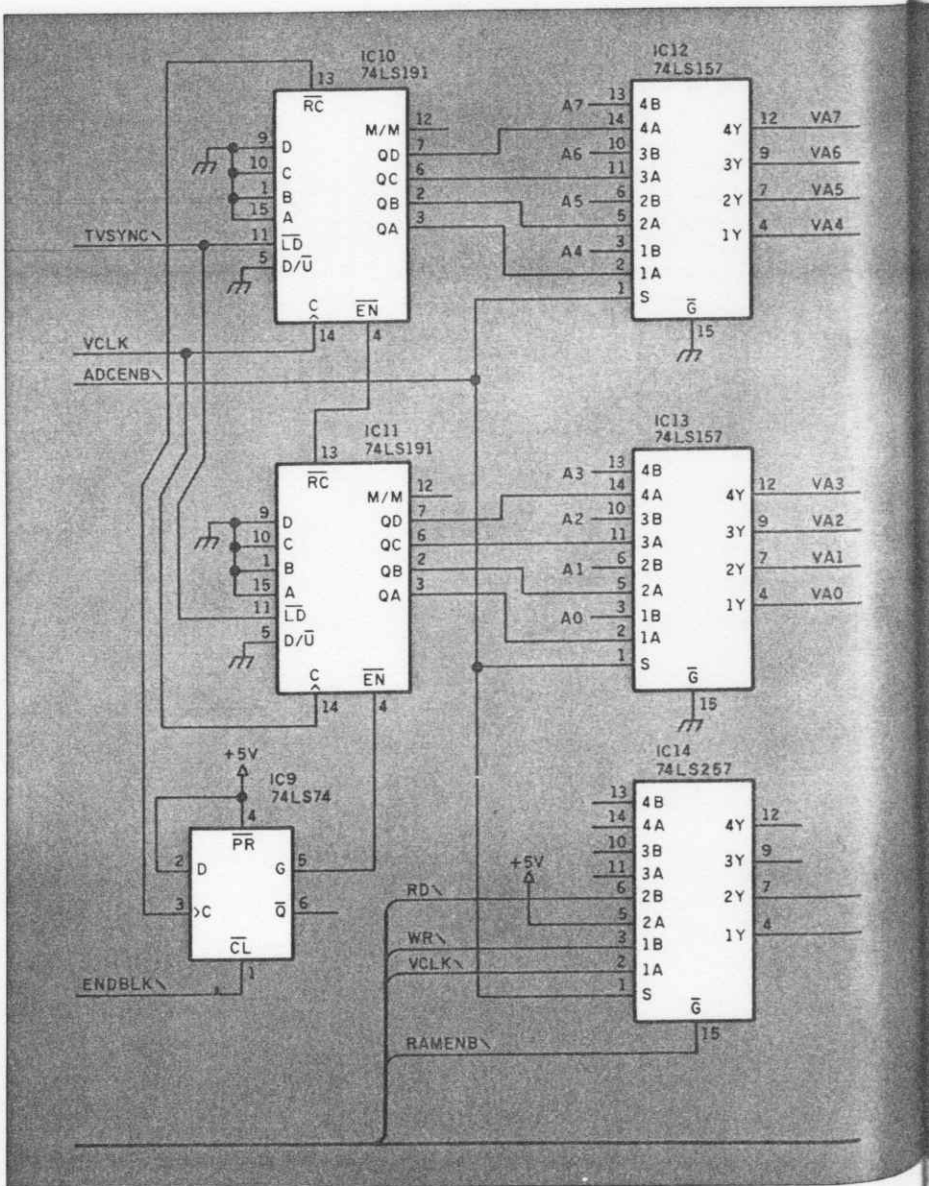


Figure 1: *Continued.*

address in the upper 8 bits of the memory address. Two 74LS191 4-bit counters provide the lower 8 bits automatically. At the start of each scan line, the counters are cleared, and the 8031 sets up the line number. After each pixel has been digitized and stored, the counters increment the address. Since there are about 66 microseconds between the start of each scan line, the processor has plenty of breathing room.

The 74LS257 ensures that we are always writing to the RAM during digitization. Normal processor reads and writes can be performed at any other time for image transmission or other processing.

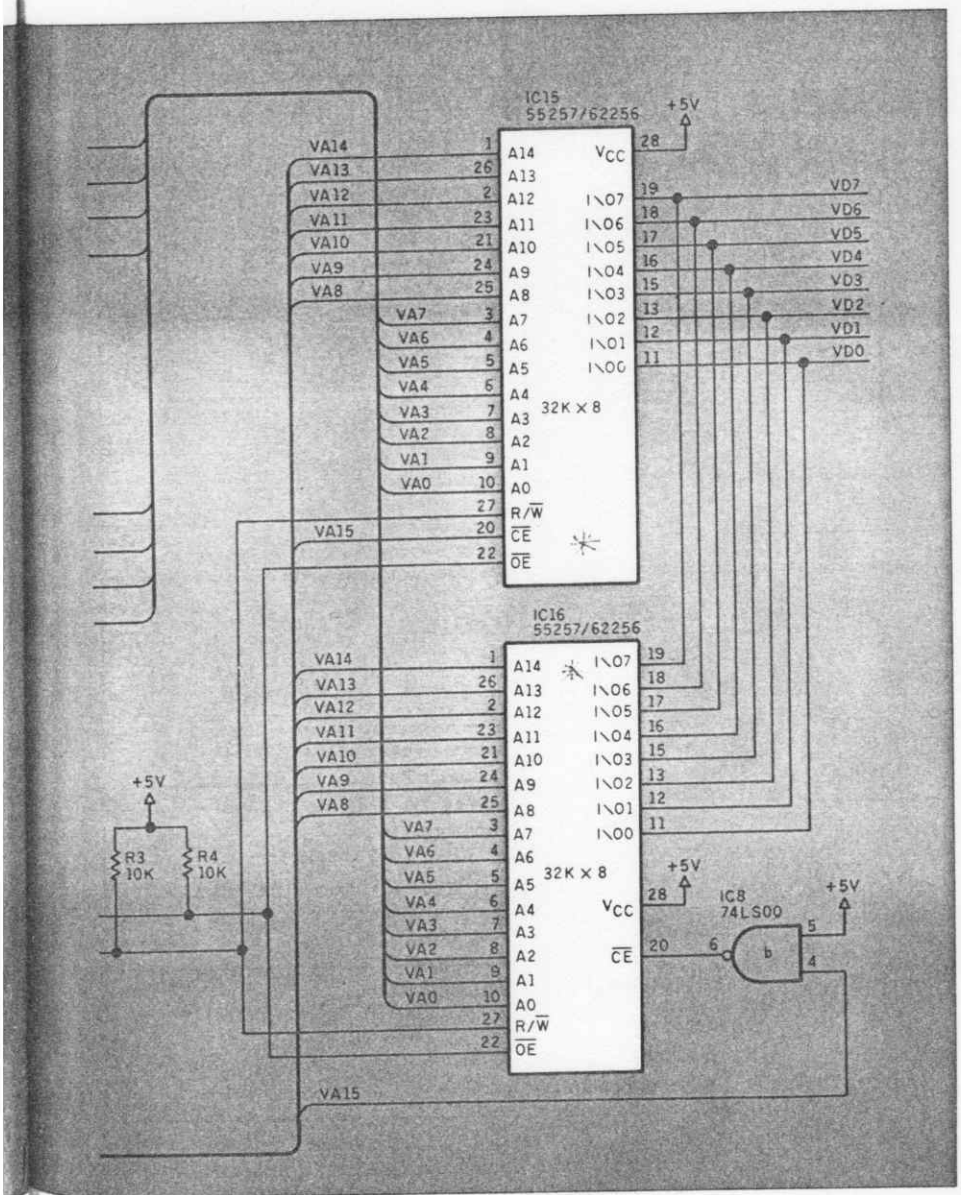
The Software Connection

The digitizer/transmitter software is a simple loop that captures a video field in the RAM buffer, then compresses and transmits the data via serial RS-232. A DIP-switch setting determines whether the digitizer/transmitter will transmit continuously or wait for an XON from the display/receiver before starting each field (see table 1).

The software begins video data capture when it detects the first vertical sync pulse in a field, as described above. The program then waits for the vertical-blanking delay (determined by using the 8031's internal timer) before enabling the RAMs and the A/D converter. Next, the program counts sync pulses and increments the RAM line address after each pulse. When the buffer is full, it disables the RAMs and the A/D converter to prevent further buffer writes.

The flowchart shown in figure 2 describes the process used to compress the video data in each line. Notice that a unique sync byte designates the start of the field and the line within the field, as well as the end of the video data.

The software captures a video field, then it compresses and transmits the data.



Getting Started

Assuming that you have an ImageWise display/receiver in good working order from last month's project, use the procedure that follows to make your first digitized video connection.

First, connect the camera to the monitor (a coaxial cable without any fancy hardware between) and get a picture that's well-lighted and focused. The adage about "garbage in, garbage out" certainly applies to this operation! Make sure that you've got the monitor terminated in 75 ohms.

Connect the display/receiver to the monitor (remember to disconnect the camera first). Set the DIP switches to 28.8 kilobits per second and no time-out (continuous pictures; see last month's article for a DIP-switch-setting guide for the display/receiver).

Turn the video level (R8) trim potentiometer to midrange and then plug in the power. Do not connect the digitizer/transmitter. The display/receiver will display a diagonal test pattern that includes a gray scale ranging from full white to full black. Adjust the video level trim potentiometer so that the monitor shows the complete range of shades. You may have to tinker with the monitor's hold, brightness, and contrast controls.

Connect the camera to the digitizer/transmitter and install the 75-ohm terminator jumper (JP1). If you're using a color camera, install the color filter jumper (JP2); otherwise, remove it. Set the DIP switches to

28.8k bps
16-line vertical delay
Compression enabled
Ignore +/−1 count changes
Paced mode disabled

Turn the delay (R21), black level (R18), and white level (R14) trim potentiometers to midrange; connect a serial cable between the digitizer/transmitter and display/receiver; and plug in the power. (Note: Set up the serial cable so that pins

continued

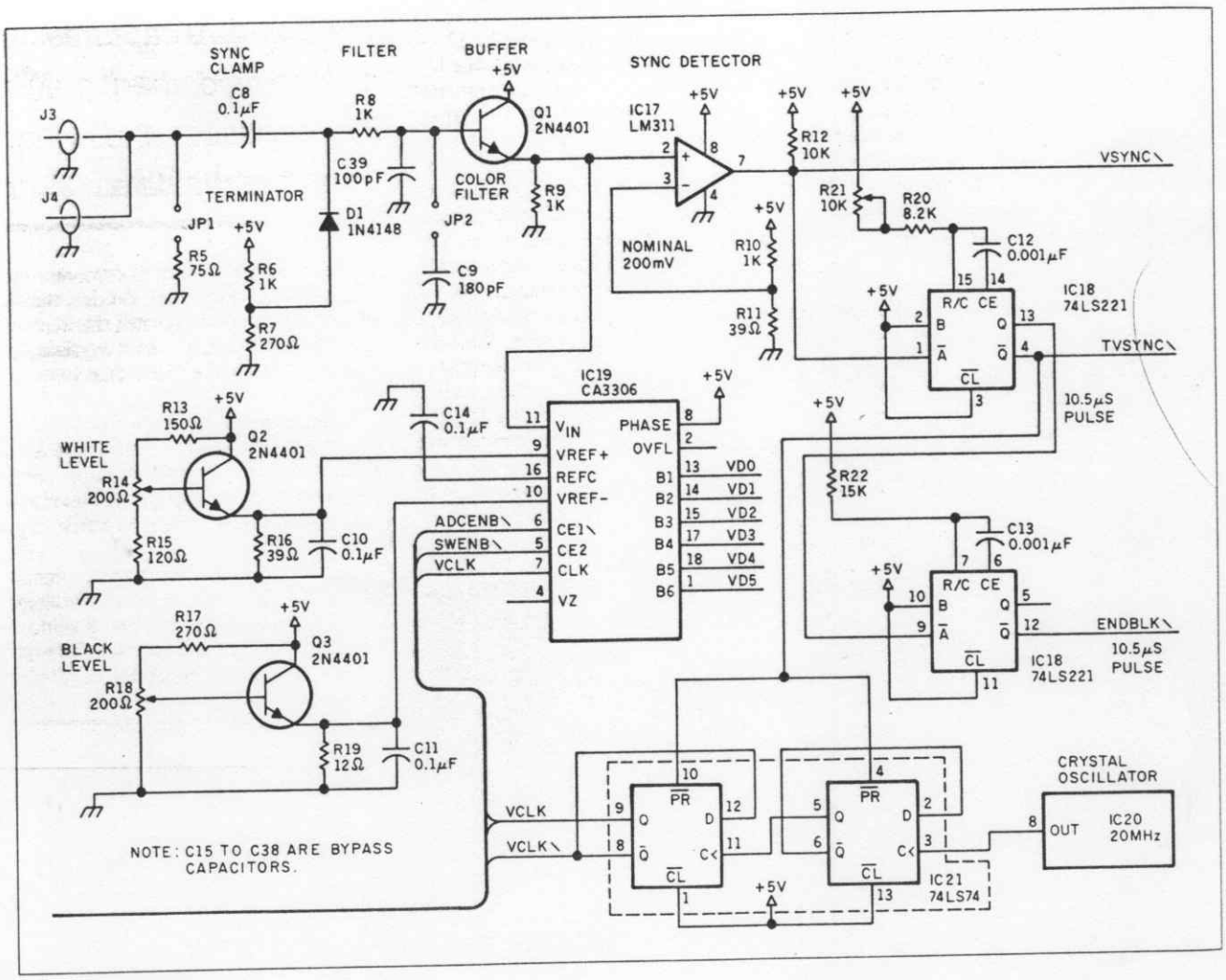


Figure 1: *Continued.*

Table 1: *ImageWise digitizer/transmitter DIP-switch settings. ON and OFF refer to switch positions.*

Switches 1, 2, and 3 select the serial bit rate (must match the receiver's rate).

1	2	3	Serial bit rate (must match receiver rate)
OFF	OFF	OFF	300 bps
OFF	OFF	ON	600 bps
OFF	ON	OFF	1200 bps
OFF	ON	ON	2400 bps
ON	OFF	OFF	9600 bps
ON	OFF	ON	19.2k bps
ON	ON	OFF	28.8k bps
ON	ON	ON	57.6k bps

(Note: 4800 bps was intentionally omitted.)

(Note: Switches 7 and 8 are not used and must be OFF.)

Switch 4 selects the vertical blanking delay from the first vertical sync pulse.

4	Vertical blanking delay
OFF	16 lines (normal)
ON	20 lines (extended)

Switch 5 enables or disables picture compression.

5	Run-length encoding
OFF	disabled (no compression)
ON	enabled (compression)

Switch 6 enables or disables the +/−1 count change compression.

6	Compress +/− 1 count changes
OFF	encode (less compression)
ON	ignore (more compression)

2 and 3 are exchanged and pins 5 and 7 are straight through.) You should see the digitized picture appearing on the monitor, painting from top to bottom. After the entire scene is done, another image will be sent. You'll be able to see a horizontal line marking the descending edge of the new picture overlaying the old one.

You should now adjust the black and white trim potentiometers to get the maximum amount of detail in the picture. If the black level is too high (clockwise), dark areas will have little detail, and the whole picture will be dark. If the white level is too low (counterclockwise), the bright areas will suffer, and the picture will be light. You should have a high-contrast scene in front of the camera to make sure you have the right levels. (On the other hand, don't make the black level too low or the white level too high, because that will reduce the number of digital levels in your scene. For example, if your camera's highest voltage is 1.4 V, it does no good to have the white setting at 2.0 V; that 0.6-V difference contains some digital codes that will never be used.)

Adjust the delay trim potentiometer so that the scene is horizontally centered on the monitor. If your monitor has a great deal of overscan, it won't matter too much what the delay setting is. You can also adjust the monitor's horizontal hold knob slightly.

If a line of trash (there's no better way to describe it) appears at the top of the monitor's display, try setting the digitizer/transmitter DIP switch for 20 lines of vertical delay instead of 16. Some cameras produce a few lines of trash at the beginning of the field, and the digitizer/transmitter faithfully digitizes it. If you don't see anything, or if the trash moves to the bottom, leave the DIP switch set for 16 lines.

If you have a monitor connected to the camera (a viewfinder doesn't count), you must make sure that either the monitor is terminated or the termination jumper is installed, but not both or neither. You should terminate the device at the end of the camera cable, not the one in the middle.

You may want to try the filter jumper to see what effect it has on the scene. If you have a color camera, the filter is probably going to be essential; if you have a monochrome camera, you may simply like a softer picture. Do not confuse the effect of the filter jumper with the output of an unfocused camera; make sure the scene is crisp to start with.

Try flipping the digitizer/transmitter switch that ignores +/−1 count changes

continued

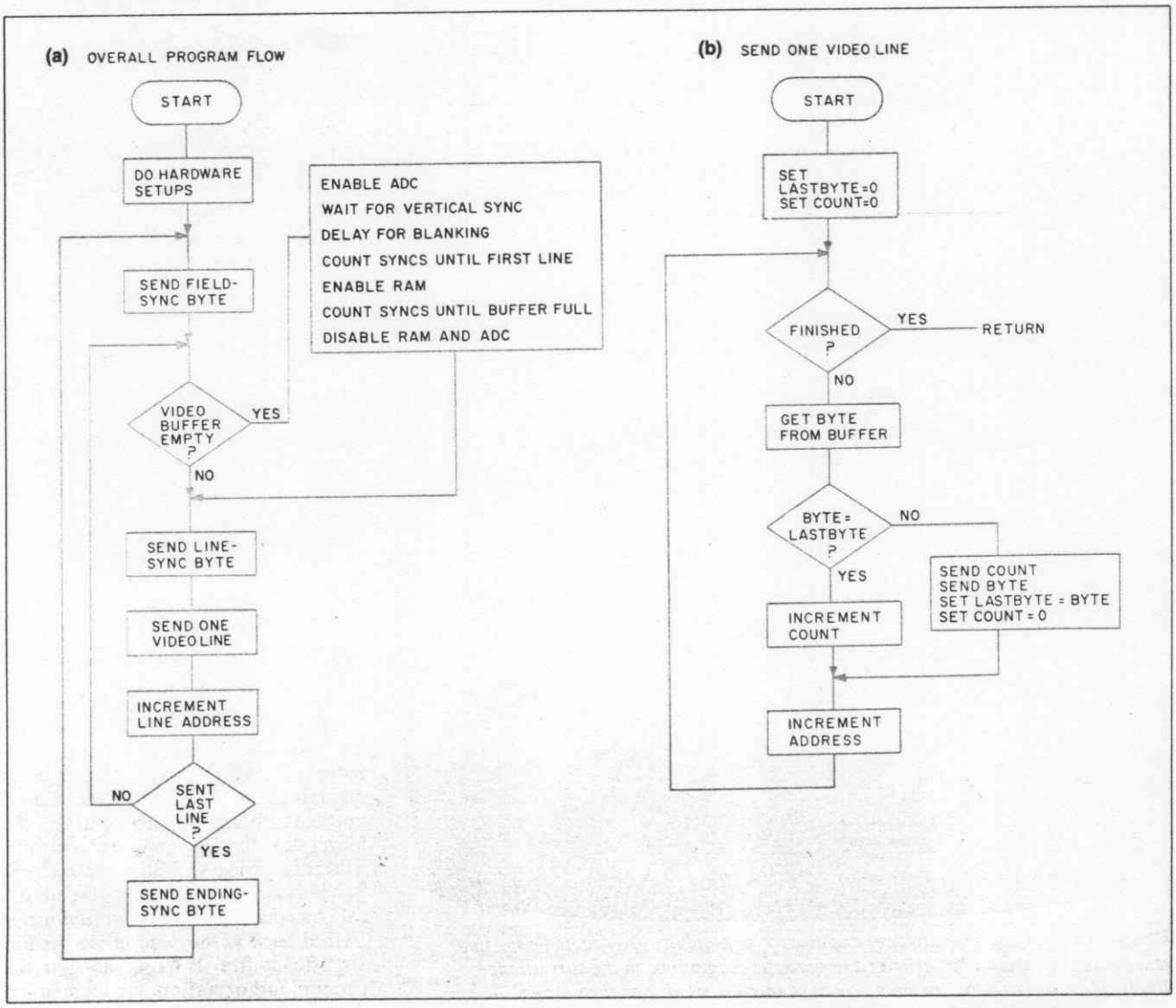


Figure 2: *Flowchart of the digitizer/transmitter software.* **(a)** *This outlines the overall flow of the software.* **(b)** *A flowchart of the routine to transmit a single video line.*

to see what happens to the flat areas in the picture. Disable compression and measure the increase in transmission time.

You can also try changing the bit rates to see which rates work for you. A direct connection can run at 28.8k bps, but for longer wires running through the house, you may need to use a slower rate. At 300 bps you can see the compression working. Remember that the display/receiver will get confused when you change the rate, so you may need to reset the system after changing any DIP-switch settings.

Cheap Buffer Option

I've described how the digitizer/transmitter works with a 64K-byte RAM field buffer made of two 32K-byte static RAM chips. It turns out that you can use an 8K-byte buffer in the digitizer/transmitter. Because the 32K-byte static RAM chips are still rather expensive, I thought it would be worthwhile to reduce the cost for applications that don't need the advantages of the full 64K-byte buffer (Note: The ImageWise kit contains 32K-byte static RAM chips.)

An 8K-byte RAM can hold 32 lines of video (8K bytes divided by 256 bytes per line). The digitizer/transmitter digitizes 32 lines, transmits them, captures another 32 lines, and so on. It keeps track of the last digitized line and starts with the next line for the next group. When it's done, it begins again with the first video line.

Using an 8K-byte RAM is simply a matter of plugging it into the first RAM socket. The 8031 program "feels around" after a reset to determine the RAM size during initialization and uses the proper addresses automatically; no DIP-switch settings are required.

The only difference between using 8K- and 64K-byte RAM buffers occurs when the scene contains moving objects. The 64K-byte buffer can hold a complete frame that is captured in 1/60 second, and it is the only configuration that can legitimately be called a frame grabber. The 8K-byte buffer holds 32 line groups that are digitized several seconds apart, so it's possible to get confused images. If you've ever seen those "panoramic" shots of a line of people with the same person at both ends of the line, you'll recognize the problem right away. But if your application doesn't involve rapid motion, you can save some money on buffer RAMs.

This trick doesn't work in the display/receiver because it must have the entire picture in RAM at all times. Replacing the 64K-byte buffer with an 8K-byte buffer would simply give you 32 lines' worth of picture.

Table 2: *Hayes Smartmodem DIP-switch settings for use with the ImageWise system.*

Switch	Setting	Description
1	DOWN	Smartmodem DTR input on pin 20 is forced active.
2	UP	Don't care, UP for English responses to commands.
3	UP	Suppress responses to commands.
4	DOWN	Do not echo characters.
5	UP	Smartmodem will auto-answer on first ring.
6	UP	Don't care, UP for CD when carrier detected.
7	UP	Single-line phone connection.
8	UP	Disable Smartmodem command recognition.

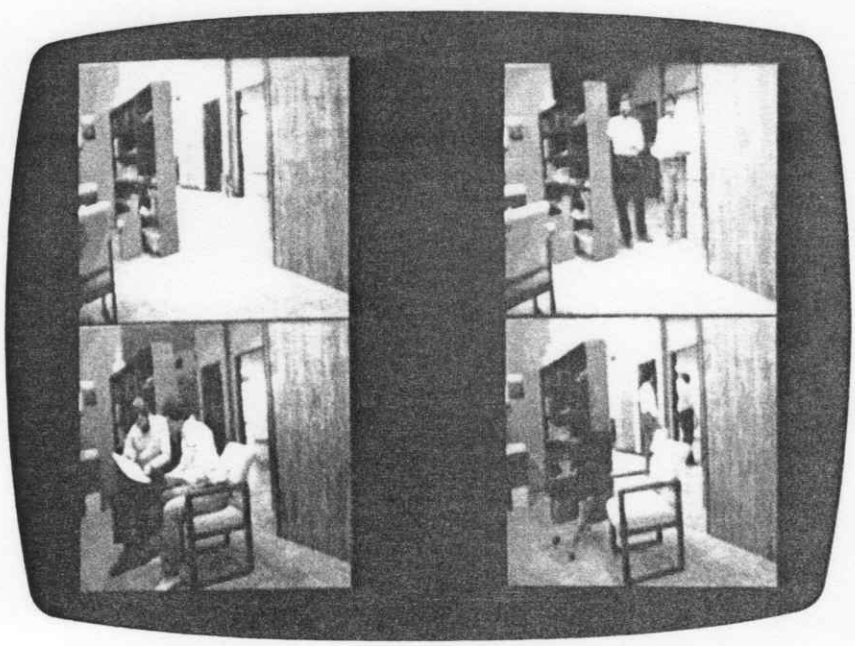


Photo 2: *The ImageWise digitizer captures a high-quality gray-scale image that can be used in areas like security and pattern recognition. In the surveillance application shown here, an empty room is entered, used, and then exited. To save space in this presentation, all four scenes are displayed on a GT180 (in 16-level gray scale).*

Using a Modem

Because both the digitizer/transmitter and the display/receiver use a two-wire serial interface and XON/XOFFs to control data flow, they can be connected through a pair of modems as well as by direct wiring. The only problem is the low data rate that modems can handle. This is an ideal application for 2400- and 9600-bps modems.

One application might be to have the ImageWise digitizer/transmitter wait for a phone call, auto-answer, and then transmit the picture it sees at that remote location. Voilà, video security system or videotelephone. The switch settings for a Hayes Smartmodem 1200 are shown in table 2. Note that you can use other modems as long as they have auto-answer capabilities.

Using ImageWise

The ImageWise system can be used as a digitizer and display board pair or as individual components of some higher-function system. Used as a pair, "teleimaging" becomes a reality. By adding the sense of sight to our ordinary audio communication, we add a new level of communication and understanding. No longer does the field engineer have to be frustrated trying to justify replacement rather than repair of an expensive electrical component. A quick digitized image flashed back to the head office verifies significant fire damage and gets the proper authorization for immediate action.

The key factor in this new level of communication is the old saying that "seeing

is believing." Consider another example: You are a consultant at a customer's site, and some question arises as to the actual wording and the date of a revision note on an important schematic. Rather than waste a day with express delivery, you can call your office and have them transmit a digitized image of the portion of the schematic in question complete with the authorization signature and date as they appear.

ImageWise has an infinite number of stand-alone uses. It can be used to instantly communicate fingerprints and ID photos, monitor traffic at remote intersections, monitor remote security risk zones (see photo 2) via auto-answer modems, aid in conducting company-wide lectures (standard audio teleconference with pictures of the blackboard periodically sent to all locations; see photo 3), or send x-rays and CAT scan pictures to medical personnel for corroborating diagnosis.

Some of these uses might seem ambitious for ImageWise, but similar more expensive units are already being applied in these areas. My immediate application may seem mediocre by comparison. Recently, I've been spending time out of the Circuit Cellar at an office across town. Since I already had a TV camera in my driveway (no windows in the cellar, remember), I simply attached it to the digitizer/transmitter and a 2400-bps auto-answer modem. Now I can call the house and get the latest snapshot or simply leave it on all the time as a real-time display of all the activity around the house (see photo 4). (I have four telephone lines; I can call the Home Control System on its own line if I want to have some real fun with someone like a delivery man. These guys all think my house is haunted.)

You'll note that I have described ImageWise only in terms of a 256- by 256- by 6-bit digitizer. Because I intended to use it with a modem, I felt the need to increase the picture-transmission speed. One way is to reduce the resolution to 128 by 121 or 64 by 61 bits. Even though such resolutions produce grainy images, they are still quite recognizable, especially if they are of familiar faces or geography (the recently advertised Mitsubishi video telephone has a 94- by 94- by 4-bit resolution by comparison). The 64- by 61-bit image is transmitted eight times faster than a 256- by 256-bit image and is suitable for monitoring gross changes in a driveway scene when a car or a person approaches.

When something appears, I can immediately change the DIP switches on the display board for a higher resolution and trigger another picture while the form is still in view. (The frame is grabbed instantly and is independent of the transmission time.) The picture-repeat rate and resolution, remember, are commanded from the receiver and not fixed by the transmitter. The interaction is completely dynamic. My next activity is to connect the ImageWise digitizer to a computer and let it decide what's happening for itself and make all the decisions.

Fortunately, this is as easily said as done. Probably the most significant feature of ImageWise is that it is computer-nonspecific. It is a serial RS-232 I/O device that does not depend on any computer-specific bus. The ImageWise digitizer/transmitter's serial port can be connected to any personal computer. The computer can receive image data and store it on disk or send it to a similarly connected display/receiver board. So far we've written the software for my SB180 and the IBM PC. Others will follow.

Experimenters

While printed circuit boards and kits are available for the ImageWise system, I encourage you to build your own. If you don't mind doing a little work, I will support your efforts as usual. A hexadecimal file of the executable code for the 8031 digitizer and display system EPROMs, sample picture files, and the Turbo Pascal

continued

Photo 3: *This is how a picture of me standing in front of a blackboard in Connecticut would be received by an ImageWise display/receiver board in California. A video printer like the Mitsubishi unit shown here can save the current scene while another is being transmitted.*

Photo 4: *The equipment counter in the Circuit Cellar where I took most of the photos with the setup you see. The interesting point to realize here is that this picture is completely digitized. A video camera (out of view) is pointed at the two stacked monochrome monitors and the film camera. The monitor on the bottom displays the picture as it is produced by the video camera and input to the digitizer/transmitter board. The display on top shows the output of the display/receiver board after it receives the data from the digitizer. It is the object of view by the film camera. This picture is the screen of the top monitor.*

code for storing images on an IBM PC are available for downloading from my bulletin board at (203) 871-1988 (similar code written for the SB180 in machine language is also available). Alternatively, you can send me a preformatted IBM PC or SB180 disk with return postage, and I'll put all the files on it for you (the hexadecimal file could be used with my serial EPROM programmer, for example). Of course, this free software is limited to noncommercial personal use.

Next Month

Once you've got a picture in digital format, you can write programs that perform magic tricks with it (hardware people like to think that way). By manipulating the binary data making up the picture, you can transform it into another picture that may be more meaningful. You can even combine two pictures to find differences—this is called image processing.

Now that we have the ImageWise digitizer, we have the means to perform some real experiments. I know many tutorial articles on image processing have been published, but the true Circuit Cellar creed is to build it yourself. Using ImageWise, next month I'll demonstrate how the basics of picture comparison, enhancement, and other image-processing fundamentals can be done for real. Finally, in another conversion of tutorial to example for August, I will demonstrate the process of colorizing the 64-level gray-scale ImageWise picture. ■

Special thanks to Ed Nisley for his expert collaboration on this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, P.O. Box 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

CCI
P.O. Box 428
Tolland, CT 06084
(203) 875-2751

1. ImageWise digitizer/transmitter board experimenter's kit. Contains digitizer/transmitter printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with transmitter software, and CA3306 flash A/D converter and manual with complete parts list.
DT01-EXP $99

2. ImageWise display/receiver board experimenter's kit. Contains gray-scale display/receiver printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with receiver software, Telmos 1852 video D/A converter, manual with complete parts list, and an IBM PC 2.0 disk containing sample digitized images and test patterns.
DR01-EXP $99
DT01-EXP and DR01-EXP together $179

3. ImageWise digitizer/transmitter full kit. Contains all digitizer/transmitter components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, CA3306 flash A/D converter, manual, and IBM PC 2.0 disk containing utility routines for storing and displaying (dot-dithered, not gray scale) and downloading image files using an IBM PC. Does not include power supply or case.
DT01-KIT $249

4. ImageWise display/receiver full kit. Contains all gray-scale display/receiver components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, Telmos 1852 video D/A converter, manual, and an IBM PC 2.0 disk containing sample digitized images and test patterns. Does not include case or power supply.
DR01-KIT $249
DT01-KIT and DR01-KIT together $489

ImageWise is also available assembled. Call CCI for source and availability of assembled boards and complete systems, black-and-white TV cameras, 32K-byte static RAM chips, and power supplies. Software utilities are also available in SB180 format.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Surface delivery (U.S. and Canada only): add $3 for U.S., $6 for Canada. For delivery to Europe via U.S. airmail, add $10. Three-day air freight delivery: add $8 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express). Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

To be included on the Circuit Cellar mailing list and receive periodic project updates and support materials, please circle 100 on the Reader Service inquiry card at the back of the magazine.

Steve Ciarcia

Part 1: Image Processing

Using the ImageWise Video Digitizer

This digitization and display process is easy to duplicate

While I was writing the second part of the ImageWise project, a BYTE editor sent me copies of the image processing theme articles used in the March 1987 issue. After I got over my first reaction to the common thread of the articles (it's almost all software—yech!), I realized that, while I was covering the hardware specifics of ImageWise more than adequately, to do real justice to the subject I should include more on using and processing the data created from the digitizer.

Getting from here to there constituted a problem, however. While many people can read and instantly visualize the image transformations described in these image processing tutorial articles, some people prefer an alternative approach to such presentations. Although the March issue was devoted to image processing, I'd like to think there is a difference when I discuss a subject.

I describe ideas, but I also try to include a little hands-on experience. Unlike a tutorial that contains little mention of the hardware you might use to duplicate such feats, all of the picture data used in this article was digitized on the ImageWise digitizer/transmitter and displayed on its companion display/receiver board (except for the zoom shots, which are displayed on a GT180). You should be able to easily duplicate the process.

I have expanded the original two-part ImageWise hardware project to include two more articles with a little software. Admittedly, I am out of my element, and I ask you to bear with me if I drop a few bits now and then (think of it as poetic license). I couldn't pass up an opportunity to string together such interesting ideas as image processing and colorization—which I can actually demonstrate.

This month, I will focus on image processing. As in the related tutorials, I'll take a digitized image and detect edges, enhance it, filter it, enlarge it, subtract it, and create other more useful images. Next month, I'll take the black-and-white ImageWise system and combine it with a computer to demonstrate a little homegrown movie colorization.

First, a quick hardware review of ImageWise will show you what the data is that I am processing (see my May and June articles for more details).

Picture Format

The ImageWise digitizer/transmitter digitizes a single field of the camera's video signal on-the-fly, converting it into 244 rows of 256 pixels each. The rows are numbered from 0 to 243, and the pixels are numbered from 0 to 255 in each row.

A pixel's brightness is represented by one of 64 gray levels, with a black pixel equal to 0 and a bright white pixel equal to 63. Each pixel requires 1 byte of storage, so there are 62,464 pixel bytes per image. Software adds some additional control-information codes to simplify the display/receiver's job, giving a total of 62,710 bytes in an image.

The digitizer/transmitter compresses the video data using run-length encoding to reduce the time needed to send it over the RS-232 serial link. When the digitizer/transmitter finds a gray-level value repeated more than twice in adjacent pixels (a "run") in the line, it replaces the repetitions with a count. Typical scenes are reduced by 50 percent to 75 percent, with a corresponding speedup in transmission.

The display/receiver accepts RS-232 data, decompresses it into a RAM display buffer, and generates the synchronization signals required to show the images on a standard composite-video TV monitor. The result is a TV picture that looks remarkably like the original scene.

The Personal Computer Connection

Because both the digitizer/transmitter and display/receiver communicate over a standard RS-232 line, you can connect either one to a serial port on a personal computer (the unit can connect to any computer with a serial port, but all my examples use an IBM PC). When the computer is connected to the digitizer/transmitter, it acts as a display/receiver, storing the image data on disk. When it's connected to the display/receiver, it acts as a digitizer/transmitter and sends the stored images out for display.

The computer can accentuate or suppress details in an image by performing simple arithmetic on the numeric values for the pixels. For example, a program can compare two scenes by subtraction, and a count of nonzero values in the result can tell you whether something has moved into (or out of) the picture.

This article demonstrates a tool kit of programs that you can use to develop a complete image processing application. The programs are written in Turbo Pascal for an IBM PC, but you can easily convert them for use on other computers. I used an 8-megahertz IBM PC AT with 640K bytes of RAM, a 10-MHz 80287 math coprocessor, and a 1.2-megabyte RAM disk to develop these programs. They will work on any computer that runs

continued

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. The author of several books on electronics, he can be reached at P.O. Box 582, Glastonbury, CT 06033.

Turbo Pascal and has sufficient RAM (about 512K bytes) but might take somewhat longer to run. Because the images are displayed on a TV monitor connected to the display/receiver, you don't need a graphics display on the computer.

Serial Setup

You set the data rate on the serial link using DIP switches on the digitizer/transmitter and display/receiver boards. Although the maximum data rate is 57.6k bits per second, the PC simply can't keep up at that rate with the present software. While I could have used some computer assembly code to tweak the critical loops, I felt it was better to use a more easily understood technique. So the programs are limited to half the maximum rate: 28.8k bps. If your computer can't handle this rate, you must recompile the programs to use a lower rate.

Only two programs actually communicate with the ImageWise boards. The Grab program prompts the digitizer/transmitter to send an image and stores it on disk. The Show program reads the disk file and sends it to the display/receiver. Both use the COM1 serial port, so you'll have to swap cables when you use each program. (I used a serial-port switch box, but you can easily recompile the programs to grab images from COM1 and show them on COM2.)

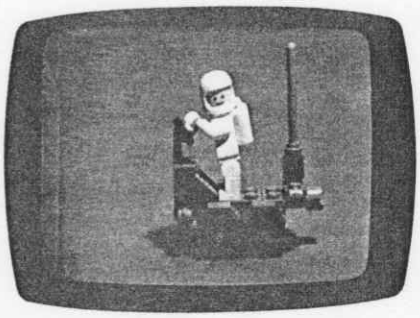


Photo 1: *An image captured by the ImageWise digitizing system.*

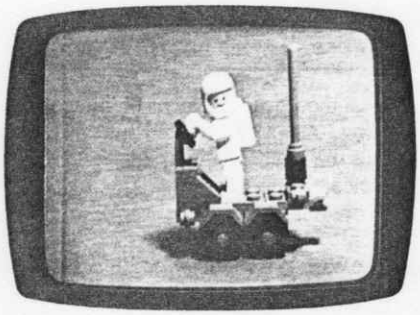


Photo 2: *I created this image by multiplying the pixels in photo 1 by 2 using the Multiply program.*

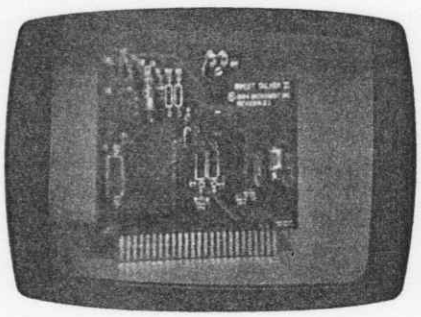


Photo 3a: *The digitized image of a circuit board.*

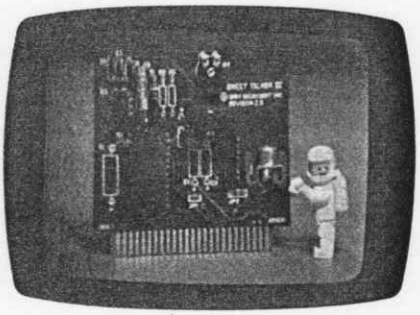


Photo 3b: *I have added something new.*

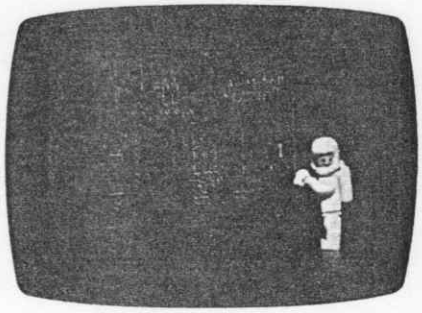


Photo 3c: *You can use the Subtract program to discover what has changed.*

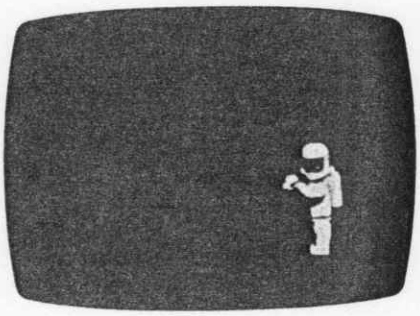


Photo 4: *You can run the Thresh program on the image in photo 3c to remove background clutter.*

One of the first tasks I have is undoing one of ImageWise's features. Although the compressed data format reduces the transmission time, it's not well-adapted to image processing. The programs must examine every image pixel, something that's not easily done with run-length-encoded data. So Grab decompresses the images before it stores them on disk, creating a 62,720-byte file for each picture. There are 62,710 image and control bytes, with 10 padding bytes added to fill out the file's last 128-byte block.

The Show program and the display/receiver can handle either run-length-encoded or expanded files, so there's no problem sending them to the display/receiver, except for the increased transmission time. The Compress and Expand programs convert between the two formats.

Taking Pictures

In addition to the ImageWise digitizer/transmitter and display/receiver boards, you'll need a TV camera and monitor, a tripod for the camera, and some RS-232 and video cables. A color TV camera will work fine, even though the digitizer/transmitter is designed for monochrome. If you see herringbone patterns on the display/receiver, install the Filter jumper on the digitizer/transmitter to remove the color information from the camera signal.

A zoom lens is a great help because you can adjust the focal length to fill the screen with the scene. If you are taking pictures of small objects, you might also need a macro lens or attachment. Most consumer TV cameras come with a macro-focusing zoom lens, so you're probably in good shape if you have one.

I captured the scenes in this article using a monochrome camera equipped with a 15- to 75-millimeter zoom lens. I used a 75-watt desk lamp for illumination. The camera lens was usually opened wide to f/2.1.

The first rule of photography is to get enough light on the subject. While you can use light meters and judgment, checking the actual results is better. The Histo program analyzes an image and reports on the number of pixels having each of the 64 possible brightness levels. Figure 1 shows the output of Histo for the image in photo 1.

The large peak is created by the desktop and background areas that are all more or less the same shade. There are relatively few black areas (near 0) and relatively few white areas (near 63). The peaks on every other pixel count indicate a little bit of noise in the A/D circuits.

Notice the small number of pixels brighter than about 30. Although it's bet-

ter to increase the amount of light on the scene, you can achieve a similar result by multiplying each pixel by a constant. Photo 2 is the same as photo 1, with each pixel multiplied by 2 using the Multiply program. This is nearly equivalent to increasing the exposure by one f-stop, thus doubling the brightness.

Figure 2 is Histo's output for the image in photo 2; notice that the pixel values are all even (multiples of two) except for the pixels that "stuck" at 63. The brightest areas of photo 2 look flat because they are all the same value. Increasing the illumination would have filled in the odd-numbered pixels.

When you're setting up a new picture-taking session, always use Histo to make sure you're getting enough light on the scene. It's all too easy to twist the brightness knob on the monitor, which doesn't do anything for the digitizer/transmitter.

What's New and Different?

One of the more interesting things you can do with two images is to find the differences between them. Photo 3a shows a small circuit board, and photo 3b has something new added. By using the Subtract program to produce the image in photo 3c, you can see exactly what changed.

Often there will be minor, inconsequential differences between the images. You can see some background clutter in photo 3c resulting from small differences in lighting and position. Regardless of how careful you are, these differences will occur. What you need is a program to get rid of the irrelevant details.

The Thresh program sets pixel values below a specified threshold level to 0 (black). Running Thresh to remove all pixels below 40 gives the image in photo 4. In addition to suppressing the clutter, Thresh removed the face inside the helmet. This should serve as a reminder that Thresh is concerned only with the brightness of each pixel: Because the face pixels are less than 40, they are set to 0 just like the background clutter.

Photo 4 contains only the parts of photo 3b that aren't in photo 3a, but there are some shadows and reflections in addition to the figurine. The pixel values represent the brightness of the scene and don't "know" whether they are part of an interesting object or the background. Your image-recognition software must distinguish between the actual objects and their shadows and reflections.

Inspection Applications

I know that many of you are interested in using video for inspection, so the next example shows what's needed to compare two pictures to find differences. Since I occasionally digress, I thought inspecting printed circuit boards for missing components was a suitable example.

A critical ingredient in any inspection task is a reference standard that "looks right." All other items are compared to that standard; anything different is regarded as an error. Of course, the differences have to be visible to be detected.

The image in photo 5a is the reference circuit board (anything less than a perfect image is due to lousy lighting). Photo 5b shows a test board with one IC missing. Notice that the ICs are darker than both the board and the silk-screen print below them, but that the capacitors are lighter than the board.

The Compare function is the same as Subtract, except that it returns the absolute value of the difference. Thus, any change will show up as a bright pixel. Photo 5c shows the results of using Compare to process the images of the reference and test boards. A lot of background clutter is due to minor variations in the boards, the lighting, and positions. A simple threshold won't remove the clutter because some of it is quite bright. The trick is to know where the important

continued

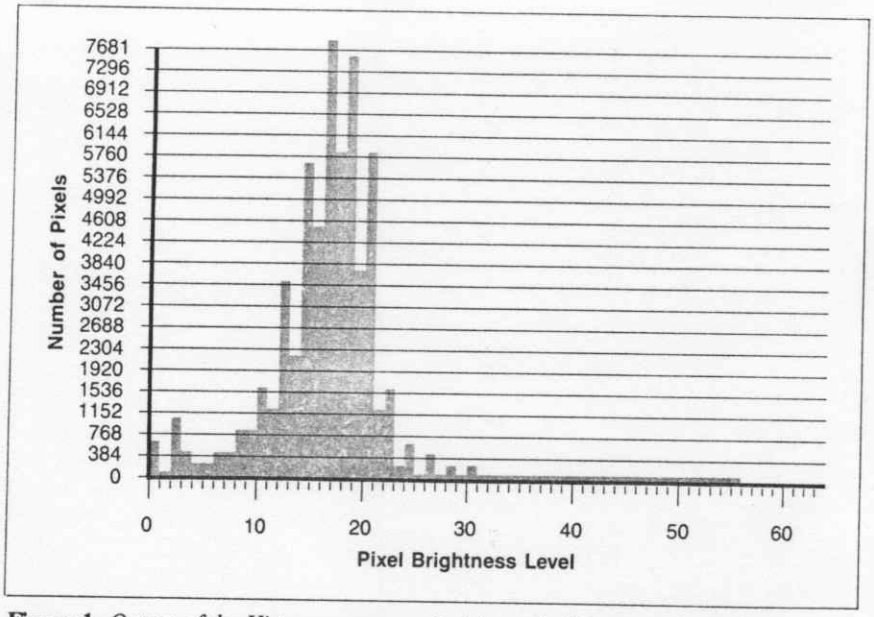


Figure 1: *Output of the Histo program: a pixel-intensity histogram for the image in photo 1.*

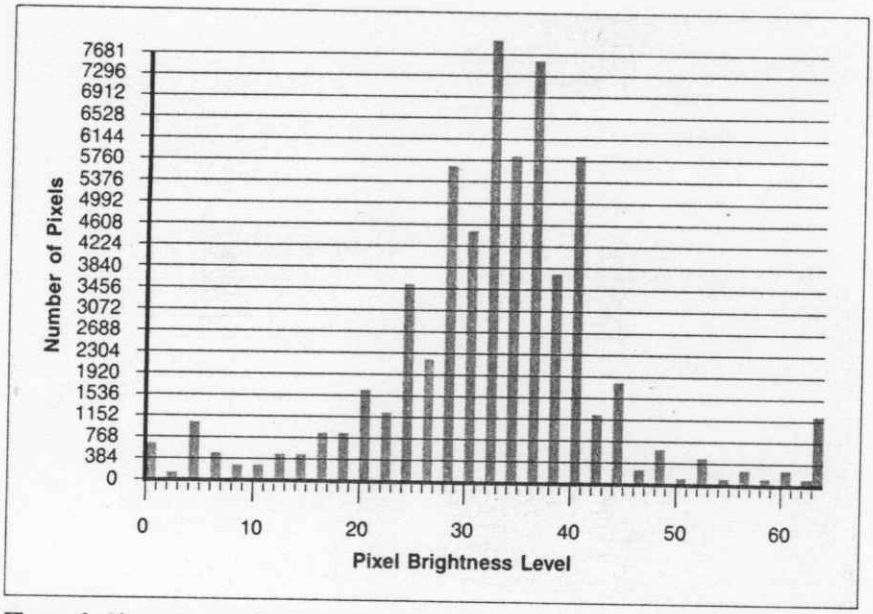


Figure 2: *If you process the image in photo 2 through the Histo program, this is what you get. Notice that all pixels are multiples of 2.*

A	B	C
D	*	E
F	G	H

* = Current pixel location

$$\text{new pixel value} = \frac{\text{Abs}(A-H) + \text{Abs}(C-F) + \text{Abs}(B-G) + \text{Abs}(D-E)}{4}$$

Figure 3: *The formula used by the Edge program.*

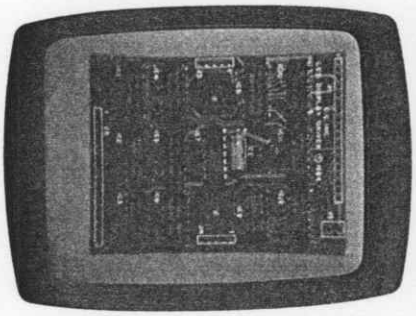


Photo 5a: *Using the Compare program, I have processed the reference image.*

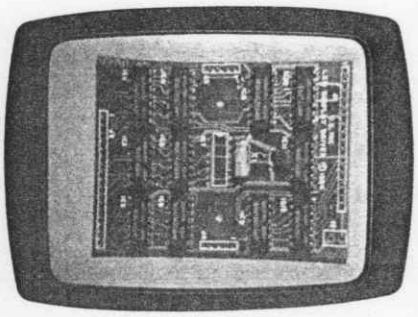


Photo 5b: *This was done with a test image from which I have removed an IC.*

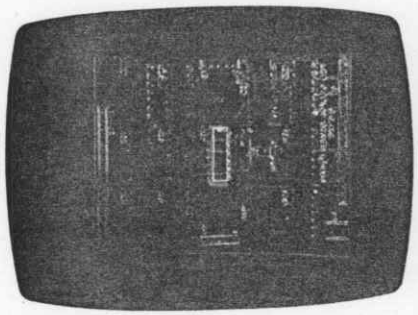


Photo 5c: *The missing IC stands out.*

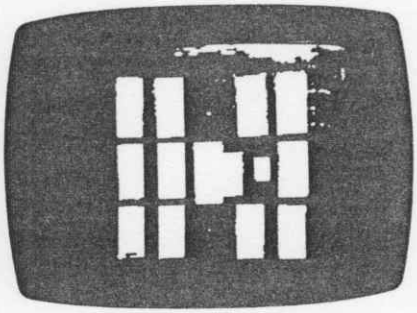


Photo 6a: *I've prepared a mask image and used it to isolate the important elements in photo 5c.*

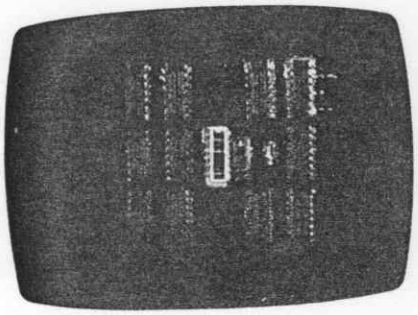


Photo 6b: *The result of processing the image in photo 5c.*

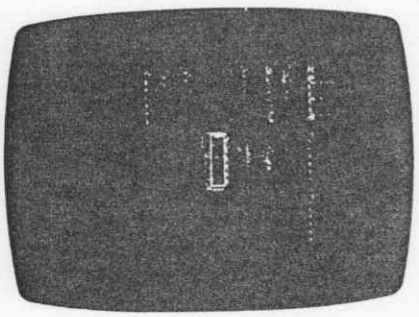


Photo 7: *Here, I use the Thresh program to remove background noise from the image in photo 6b. The missing IC in 5b now stands out clearly.*

areas of the picture are and ignore the rest.

Photo 6a shows a mask with bright areas surrounding each component location, prepared by putting white tape on a blank printed circuit board and processing the image to remove the board traces. The Mask program will suppress any pixels in one image lying outside the masked areas in a second image. The image in photo 6b is the result of using the Mask program with the image in photo 6a on the image in photo 5c.

The final step is to apply Thresh to reveal the missing IC in photo 7. What you're seeing is the white silk-screen image printed on the board. It is difficult to see the difference between a dark gray IC and a dark blue circuit board; anything you can do to increase the contrast will help. You will also need to ensure that the two images are accurately aligned and lighted to reduce background clutter. A fixture to hold the boards at an exact location relative to the camera and lights will be essential. You can use Histo, Compare, and Thresh to set up.

Edges and Filters

In some cases, you might be interested in the location of the edges of an object. For example, you might want to know that a pattern is correctly positioned without caring what color (or gray shade) it is. The Edge program produces an image that contains the difference between a pixel's neighbors, calculated as shown in figure 3. A sharp junction between a light and a dark area will result in a bright line, while a uniform area will be reduced to black. The actual shades are not important, only the differences between them.

The Edge routine is a bit more complex than Compare. It finds the absolute value of the differences between the eight pixels surrounding the current pixel in all four directions: vertical, horizontal, and the two diagonals. This is a simple example of a more complex operation called a convolution, which you can use to identify other features in an image.

I restricted Edge to a 3 by 3 set of pixels to reduce the amount of time required to get the answer: It works well enough for these examples. You might want to experiment with a 5 by 5 or larger array, which will let you identify edges more precisely, particularly diagonals at other than 45 degrees. You can also identify the direction of the edges by removing the absolute value function. An edge-detector algorithm that performs this operation (see figure 3) is

$$\text{edge} = \frac{(A-F) + (B-G) + (C-H)}{3}$$

This operation would return a positive value for the horizontal edge between an upper, bright object and a lower, dark object. Reverse the two objects, and the sign becomes negative. Because the results must be returned as pixel values, you will need to add a fixed offset before setting the final pixel value.

Usually, you will have to multiply the result of Edge by 2 or 3 to make all the edges visible on the monitor. Thresh can then suppress all the "soft" edges. Photo 8b is the result of running Edge on the image in photo 8a, multiplying by 3, then using Thresh to remove pixels below 30.

It's also possible to remove edges and textures. Filter averages four pixels using the algorithm shown in figure 4 to produce the output image. Compare the images in photos 9a and 9b to see how Filter reduces "crispness" and fine details. This can be useful if you have an object with fine detail that is not needed by the rest of the processing.

Intruder Alert!

One obvious application for image processing is in a security system that can compare two images, decide when something has changed enough to warrant human inspection, and sound an alert (or fire the laser, or whatever).

You've seen most of the pieces already:

1. Grab a reference image.
2. Grab a test image.
3. Compare the images.
4. Thresh the result to remove clutter.
5. Count the number of changed pixels.
6. If the count is high enough, take action.
7. Replace the reference image with the test image.
8. Go to step 2.

The key program is Count, which examines an image and counts the number of pixels that exceed a threshold level. The preceding image processing steps must create an image with high-intensity pixels identifying the intruder. When the count is high (corresponding to a new shape of a person on the screen), it's time to sound the alarm.

If the images don't differ by too much, you use the test image as the reference image for the next loop. This lets the system cope with small, slow changes in lighting and motion. Obviously, you could defeat this system by easing slowly into the picture, but in practice it's hard to fool.

WATCHDOG.BAT (see listing 1) combines the programs we've used so far to automate that process. The batch file can

continued

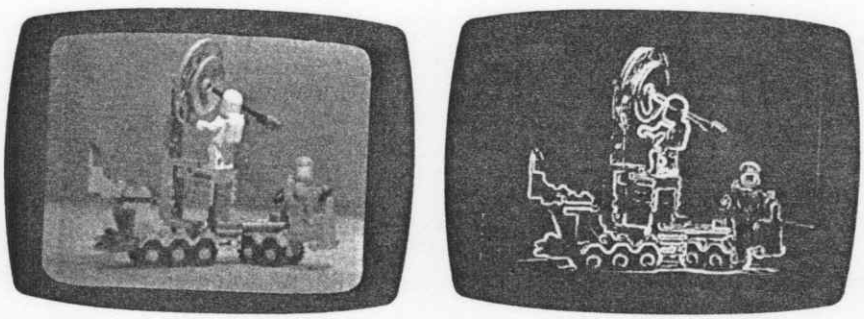


Photo 8: *Edge detection with ImageWise. The image in* **(a)** *is processed through Edge, Multiply, and Thresh to produce* **(b)**.

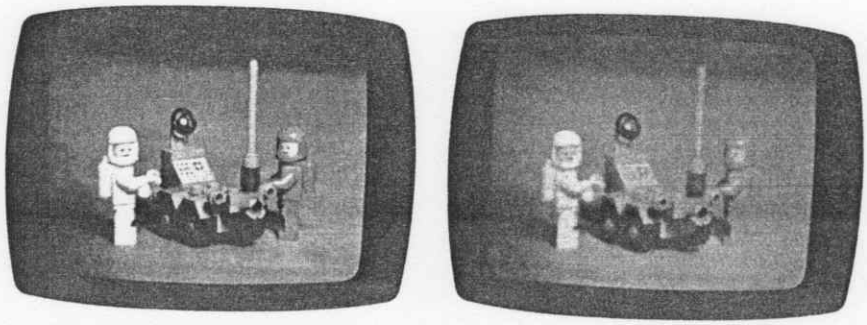


Photo 9: *Running the Filter program on* **(a)** *produces the image in* **(b)**.

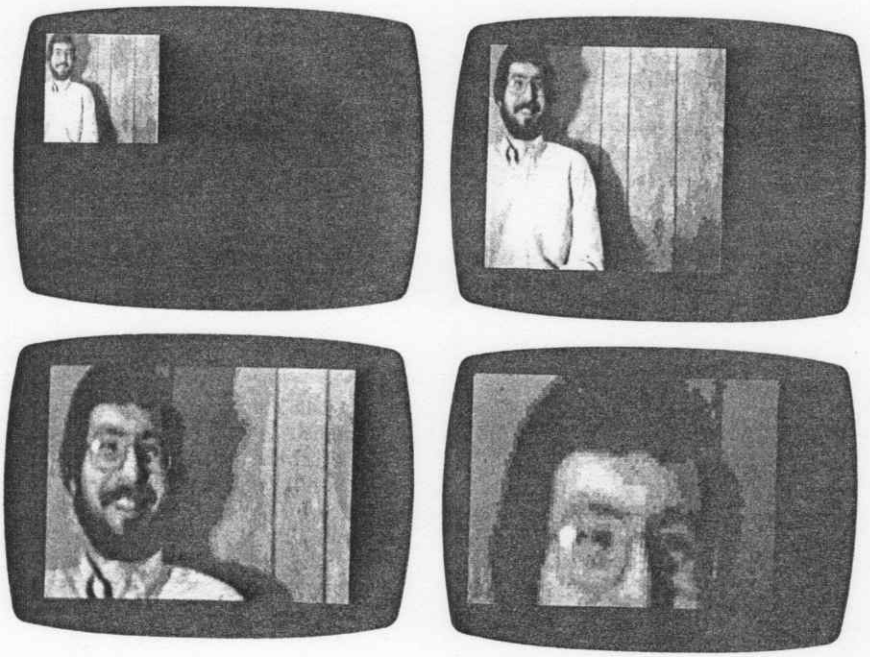


Photo 10: *An image digitized on the ImageWise digitizer/transmitter is displayed on the GT180* **(a)**. *This image is magnified 2 times* **(b)**, *4 times* **(c)**, *and 8 times* **(d)**.

A	B	C
D	*	E
F	G	H

* = Current pixel location

$$\text{new pixel value} = \frac{B + D + E + G}{4}$$

Figure 4: *The formula used by the Filter program.*

Image Processing Routines

The following is a list of the programs described in this month's column, plus some additional image processing software you might find interesting. File specs between angle brackets are optional. Results will be stored in the first file spec if the target file spec is omitted. An *n* indicates a numeric value. These programs are available from the Circuit Cellar BBS, BIX, and BYTEnet.

ADD.PAS pic1 pic2 <pic3>
Function: pic3 = pic1 + pic2.

COMPARE.PAS pic1 *n* <pic2>
Function: pic2 = pic1 if pixel > = *n*
= 0 otherwise.

COMPRESS.PAS pic1 <pic2>
Function: pic2 is the run-length-encoded version of pic1. Compressed files cannot be used by the other programs.

COUNT.PAS pic1 *n*
Function: DOS ERRORLEVEL variable = number of pixels > = *n*.

DUMPER.PAS pic1
Function: Produces formatted print dump of pic1 for hand analysis; use redirection to send output to a disk file.

EDGE.PAS pic1 <pic2>
Function: pic2 contains edge-intensity information from pic1.

EXPAND.PAS pic1 <pic2>
Function: pic2 is the non-RLE version of pic1. Expanded files are required by the other programs.

FASTDOG.PAS *n*1 *n*2
Function: Watches a scene, reports an intruder when *n*2 changed pixels exceed *n*1.

FILTER.PAS pic1 <pic2>
Function: pic2 is a low-pass filtered version of pic1.

GRAB.PAS pic1 /*n* /c
Function: Accepts picture from transmitter board and stores the expanded data in pic1. Switch /*n* prevents showing the picture on the receiver; switch /c stores the image without expanding it.

HISTO.PAS pic1
Function: Displays a pixel-intensity histogram for pic1.

INVERT.PAS pic1 <pic2>
Function: pic2 = 63 − pic1.

MASK.PAS pic1 pic2 <pic3>
Function: pic3 = pic1 if pic2 > 0
= 0 otherwise.

MULTIPLY.PAS pic1 *n* <pic2>
Function: pic2 = pic1 * *n*.

SHOW.PAS pic1
Function: Sends pic1 to the display board.

SUBTRACT.PAS pic1 pic2 <pic3>
Function: pic3 = pic1 − pic2.

THRESH.PAS pic1 *n* <pic2>
Function: pic2 = pic1 if pic1 > = *n*
= 0 otherwise.

Listing 1: *The WATCHDOG.BAT program.*

```
ECHO off
REM Syntax is:
REM   WATCHDOG brightness pixels
REM brightness is COUNT's threshold level
REM pixels is # of pixels >= brightness, in units of 100
REM   WATCHDOG 10 4
REM will alarm when 400 pixels or more are brighter than 10
REM runs best with image files on a RAM disk!
ECHO Make sure serial cable is connected to transmitter
PAUSE
:newref
GRAB ref /n
:newtest
GRAB test /n
COMPARE ref test deltas
COUNT deltas %1
IF errorlevel %2 goto gotcha
ECHO no intruder so far...
ERASE ref
RENAME test ref
GOTO newtest
:gotcha
ECHO --- Intruder alert!!! ---
ECHO Switch serial cable to receiver for display
PAUSE
SHOW test
ECHO Switch serial cable to transmitter
PAUSE
goto newref
```

examine one picture every 30 seconds or so, which might be adequate for most purposes. (If you need more speed, I have a faster program called FASTDOG.BAT in the downloadable software.)

The Count program returns the number of qualifying pixels (divided by 100) in the DOS ERRORLEVEL variable to let the IF statement decide whether an intruder is present. You should replace the ECHO statement with a program that does something useful, like turn on the lights, sound a loud alarm, or whatever you choose.

You'll need to do some experimentation to pick the best values for the threshold and count levels. Count can't tell the difference between one large change and several smaller ones, nor can it decide what the change "looks like." You'll have to mask areas of the picture or pick compromise values that don't generate too many false alarms but still never miss a real intruder. Put on your skulking suit and try to fool it.

Hardware Image Processing

While we generally think of image processing solely as software-dependent tasks, many of the newest graphics-display chips incorporate some of these functions in hardware. Most prominent

among such features is the hardware zoom or image-expansion function. The GT180 color graphics board I presented in the November 1986 Circuit Cellar has a hardware zoom that can expand an image up to 16 times.

Photo 10a shows a standard-resolution 256- by 244-pixel picture (no, it's not me this time) digitized on the digitizer/transmitter board and displayed in 16-level gray scale on a GT180 high-resolution graphics-display board. Because the GT180 has a resolution of 640 by 480, the lower-resolution digitized picture fills only the top left corner, but it expands to fill and then overflow the screen as it is zoomed. Photo 10b is 2 times magnification, photo 10c is 4 times, and photo 10d is 8 times the original image.

Conclusions

As anyone who owns a TV camera can attest, video is fascinating. Until now, small computer users haven't been able to work with pictures of the real world because the video hardware was frightfully expensive. With the hardware and software I've provided, you can take digital pictures, enhance them to pick out interesting objects, and save them for later. I'm sure you'll find many more ways of tweaking the video.

The complete source code for all the programs described in the text box on page 118 is available from the Circuit Cellar BBS, BIX, and BYTEnet.

Next Month

I'll throw some more gas on the fire as I take an old black-and-white photo and make it color. ■

Special thanks to Ed Nisley for his expert collaboration on this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, P.O. Box 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

CCI
P.O. Box 428
Tolland, CT 06084
(203) 875-2751

1. ImageWise digitizer/transmitter board experimenter's kit. Contains digitizer/transmitter printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with transmitter software, and CA3306 flash A/D converter and manual with complete parts list.
DT01-EXP $99

2. ImageWise display/receiver board experimenter's kit. Contains gray-scale display/receiver printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with receiver software, Telmos 1852 video D/A converter, manual with complete parts list, and an IBM PC 2.0 disk containing sample digitized images and test patterns.
DR01-EXP $99
DT01-EXP and DR01-EXP together $179

3. ImageWise digitizer/transmitter full kit. Contains all digitizer/transmitter components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, CA3306 flash A/D converter, manual, and IBM PC 2.0 disk containing utility routines for storing and displaying (dot-dithered, not gray scale) and downloading image files using an IBM PC. Does not include power supply or case.
DT01-KIT $249

4. ImageWise display/receiver full kit. Contains all gray-scale display/receiver components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, Telmos 1852 video D/A converter, manual, and an IBM PC 2.0 disk containing sample digitized images and test patterns. Does not include case or power supply.
DR01-KIT $249
DT01-KIT and DR01-KIT together $489

ImageWise is also available assembled. Call CCI for source and availability of assembled boards and complete systems, black-and-white TV cameras, 32K-byte static RAM chips, and power supplies. Software utilities are also available in SB180 format.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Surface delivery (U.S. and Canada only): add $3 for U.S., $6 for Canada. For delivery to Europe via U.S. airmail, add $10. Three-day air freight delivery: add $8 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express). Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

To be included on the Circuit Cellar mailing list and receive periodic project updates and support materials, please circle 100 on the Reader Service inquiry card at the back of the magazine.

Steve Ciarcia

Part 2: Colorization

Using the ImageWise Video Digitizer

Take digitized black-and-white images and convert them to color

Presuming that you aren't tired of ImageWise and the subject of video processing yet, I thought I'd stick in one more video project before I go on to other matters. Actually, I have probably picked a subject that you might know something about. The coloring of black-and-white motion pictures has been the subject of many heated debates in the entertainment industry. Movies like *The Maltese Falcon*, *Yankee Doodle Dandy*, and *Captain Blood* have been converted from their original black-and-white state to color using sophisticated computer-controlled hardware. Using computers to color these films has caused a great deal of discussion about the economic, aesthetic, and even the moral implications of modifying what many call works of art.

While colorization of movies is controversial, the technical aspects of the process are just another form of video processing and a continuation of the materials we discussed last month. With the help of your personal computer and the ImageWise video digitizer, you can experiment with colorization and form your own opinion.

This month, I will describe the software and techniques you need to add color to the black-and-white images captured with ImageWise. But before we start, let's look at the techniques Hollywood uses to color black-and-white motion pictures.

How Hollywood Does It

Converting a black-and-white film to color is a complex job that requires the talents of many people—one firm has almost 200 people working three shifts a day, seven days a week. And even with that many people involved, it takes about four months to color a full-length movie. Besides being labor-intensive, coloring a film is also expensive, with the current cost being about $3000 per *minute* of finished product. A two-hour movie costs almost $400,000 to color. While this is not cheap, it is a bargain compared to the cost of making a new film.

First, they make a print from the original film's black-and-white negative and then transfer it to videotape. This videotape is the copy they use for all work; the original film is not modified. Incidentally, the coloring process may help preserve the original picture, since old movies used nitrate film that deteriorates with age. The new print made for transfer to videotape is made using modern film stock that will be around long after the original has disintegrated.

After they transfer the film to videotape, the difficult task of determining what colors were used in the original film begins. They talk to members of the movie's cast and crew and search libraries and studio archives for color photos and set descriptions taken during the original production. The movie is also viewed to see if known objects appear in it: famous paintings, landmarks, and other recognizable objects. If the true color of an object cannot be determined, an art director will select suitable colors.

At the same time they are determining the proper colors for the film, a shot-by-shot breakdown of the entire movie is done, with a code number assigned to each scene. The significant characters and elements of these scenes are noted, and the lists are given to the coloring staff, along with the colors that are to be used. It is important that the continuity of these lists be accurate, since many people will be working on the film. A mistake can result in one group painting a car black and another group painting the same car (in a different scene) red. If these scenes are next to each other in the final film, the audience will wonder where the red car came from.

After they have determined all the colors and generated the continuity lists, it is time to begin adding color to the movie. A colorist will watch the film scene by scene in black and white and electronically add color to the picture using six colors: black, red, blue, pink, yellow, and white. These colors do not represent the actual colors that will be used but are used as indicators of light levels. Black and red represent the darkest levels of the picture, white and yellow the lightest.

The colorist breaks the picture down into individual areas or layers. An example is a shot of a hand on a desk. If the hand will be moving around the top of the desk, the colorist draws a mask around the hand to isolate it from the desk. The desk is then colored, using any colors from the entire spectrum of an electronic palette. Since the colorist colors the entire desk, even areas "beneath" the hand, when the hand moves, the portion of the desk that was under the hand will already be colored. Next, the colorist applies fleshtones to the hand. As the hand moves from frame to frame of the film, a computer will remember what colors were used and will follow the hand with fleshtones as it moves.

The colorization process uses a combination of layering and assigning colors to

continued

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. The author of several books on electronics, he can be reached at P.O. Box 582, Glastonbury, CT 06033.

I discovered that the coloring companies are using IBM PC ATs and Apple Macintoshes.

gray levels. Coloring done only by assigning colors to gray levels in each frame does not achieve the quality you have come to recognize as a professionally colorized film, but it does offer a starting point. (Since layering is extremely software-intensive and beyond our ability to duplicate here, our experiment with colorizing images will use only gray levels. The limitations of this coloring technique will be obvious, but the process is still instructive.)

After all the layers for a scene are colored and the first and last frames of a shot are completed, the colorist will select key frames that have large changes or new objects in them. The next step is to reposition the old masks and add masks for new objects as required. This is necessary when an object that was on the top layer (the hand) moves behind a new object (like a lamp). The colorist will also make any changes to the previously selected colors to allow for shadows and lighting changes. Finally, after all colors are selected and the masks are determined, a computer automatically colors all frames in the scene with the selected colors.

The colorist repeats this process for every scene in the film. The finished color scenes are then edited back together in the proper sequence. The movie is then reviewed for proper color balance and continuity. If everything checks out, the company returns the finished product to the owner for distribution.

When I started researching this article, I thought that the coloring process was probably being done with a bunch of Cray supercomputers. But after some research, I discovered that the coloring companies are using IBM PC ATs and Apple Macintoshes running custom software and connected to custom video hardware. I thought, "If they can do it with micros, it can't be too difficult, right?" Well, the key components of the coloring system are the custom software and hardware. And since at least one company has $20 million invested in its custom coloring hardware, trying to duplicate its system might be a little beyond the Circuit Cellar budget for this month.

However, even though it isn't practical for us to duplicate that company's equipment, we have built and tested some of the key ingredients in the past three months. I thought we'd apply a little journalistic license and try to use the equipment we already own. We'll just have to interpret the results in the proper frame of mind.

Setting Up

My colorization system consisted of an IBM PC with one serial port, a Genoa Systems EGA video board, an NEC MultiSync monitor, and the ImageWise digitizer/transmitter, with a camera providing a picture source (it could be a VCR, TV signal, or any such thing). The Genoa Systems EGA is a half-size board that supports IBM and Hercules monochrome, IBM Color Graphics Adapter (CGA), and IBM Enhanced Graphics Adapter (EGA) video modes.

As you'll see, we face some obvious limitations using EGA. You might wonder why I did not use PGA, VGA, or—at the very least—my own GT180 (all of which would have resulted in a more "colorful" picture). My purpose here was not to create pretty pictures that prove I understand colorization and can afford the hardware. Rather, I am trying to find the greatest audience of experimenters who might be able to recreate these experiments (this is always the purpose of the Circuit Cellar). I chose the IBM PC with an EGA display as a reasonable compromise between installed base and minimum graphics capability. Of course, if you have an Amiga or any better IBM graphics board than an EGA, you can extend the basic concept for better results.

The ImageWise digitizer/transmitter generates a 256- by 256-pixel image with 6-bit, 64-level gray scale for each pixel. Ideally, we would like to use all 64 levels in our display, but the EGA board lets you choose only 16 colors from a palette of 64. To display a picture, we have to remap the 64 gray-level values from the digitizer/transmitter into 16 values from the EGA's color palette. Fortunately, since the images are 256 pixels wide, we can conveniently use a screen resolution of 320 by 200 (we will just ignore the bottom 56 lines of the image).

The process is relatively straight-forward. You aim a camera at a black-and-white photograph or illustration. Then, using the PC-DOS "digitize and store" utility provided with the ImageWise digitizer/transmitter, you digitize the image and store it to disk. Finally, using the "coloring" software provided for this project, you select and assign color values to the black-and-white digitized images.

This coloring program is much too big to list here, but it is available for downloading from the Circuit Cellar BBS at (203) 871-1988, provided it is for your own personal and noncommercial use. Alternatively, you can send me a formatted PC disk with return postage, and I'll load it with the pertinent files. The coloring program will run on any PC with at least 256K bytes of RAM and an EGA-compatible graphics display.

The coloring program lets you assign each of 64 gray levels to one of the 16 EGA palette positions. You choose these 16 palette colors from the 64 available EGA colors. Unfortunately, it is not simply a case of substituting one of these palette colors for all instances of a particular gray level. The manipulation of the gray levels and palette colors is what colorization is all about.

As I mentioned earlier, ImageWise digitizes images into 64 levels of gray. These levels are determined by the amount of light reflected by the objects we are processing. Because we are digitizing light levels and not colors, some parts of the digitized images will have the same gray level even though they may not be the same color in the original. This problem, combined with the minimal color capabilities of the EGA, limits the results we can achieve. But these limitations are not as severe as they may first appear, and you can minimize their effects by clever manipulation of the image data.

Before you begin, you will need a subject. While you can use just about anything, some subjects will give much better results than others. Try to avoid subjects with real people in them. The EGA color palette doesn't have any colors that are acceptable as fleshtones. Also avoid subjects that are dominated by a single color. The best subjects are those with lots of colors or gray levels.

Animated cartoons are good subjects since they have a wide range of colors and have distinct borders between different color zones. I chose a scene from Walt Disney's *Snow White and the Seven Dwarfs* as the subject for my examples. Disney Studios was happy to grant permission to use this film as the subject since this is its fiftieth anniversary. I neglected to say that Snow White might be "Snow Green" when I was done.

Photo 1a shows Snow White and Dopey after being digitized and displayed in 64-level gray scale using the ImageWise system. The picture shown here, as displayed on an ImageWise display/receiver board, provided the digitized data for the EGA colorized images that were eventually generated. (Generally speaking, an IBM PC cannot display a 64-level gray-scale image without advanced graphics boards or an ImageWise

display/receiver board. However, a technique called *dot-dithering*, which creates a pseudo gray scale [e.g., like pictures in a newspaper], can be used to create a reasonable image on an EGA display. Photo 1b shows the same Snow White image in pseudo gray scale on an EGA.)

Before we begin adding color to this image, we need to decide on a method of mapping the 64 levels into the 16 colors supported by the EGA. One method of mapping the ImageWise gray levels to the EGA color palette is to ignore the 2 least significant bits of each gray-level value. You can do this by dividing each pixel data byte received from the ImageWise transmitter by four and using the integer result as the palette position. As a result, the lowest four gray levels will be mapped to palette position 0, the next four gray levels will be mapped to palette position 1, and so forth, until all 64 levels have been assigned to EGA palette positions 0 through 15. Photo 2 shows the results obtained using this method. Compare this to photo 1.

It is obvious that the colors in photo 2 are not correct. So far, we have only mapped the gray levels to palette positions with little concern for which parts of the picture are affected. The result can be red faces and blue hair. Instead, we need to select which color will be assigned to each palette position while keeping in mind which areas are painted with which colors. A somewhat more careful selection of colors (plus some fixed areas of gray, as I'll soon explain) can take the same data of photo 2 and present it with a distinct improvement, as shown in photo 3.

While the results are better, we can still make improvements. The new image has lots of gray and black pixels because the lowest 12 gray-level values received from the ImageWise transmitter are found in large numbers throughout the entire image. When we divided the gray-scale data by four, these 12 levels now occupy only three palette positions. The effect is to make large portions of the image that were distinctly different now appear equal in color.

For example, let's assume that the bottom of Snow White's dress when digitized has a gray level of 3 and the area in the upper left corner of the screen has a gray level of 1. Since our mapping scheme will assign both levels to palette position 0, the two different levels become the same color. When we try to make Snow White's dress yellow, we will also be making the upper left corner of the screen yellow. If we try to change the corner of the screen to some other color,

continued

Photo 1a: *Snow White and Dopey as digitized by ImageWise.*

Photo 1b: *The same image as displayed on an EGA 16-level gray scale.*

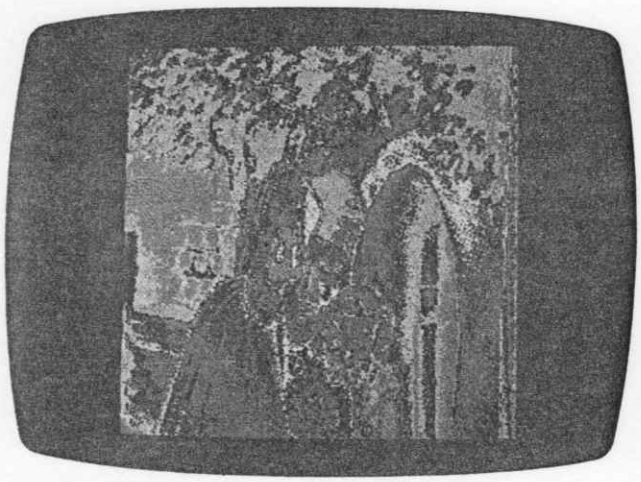


Photo 2: *A straightforward mapping of gray scale to EGA color-palette values. Each pixel in photo 1a is divided by four, and the integer result picks that pixel's color-palette value.*

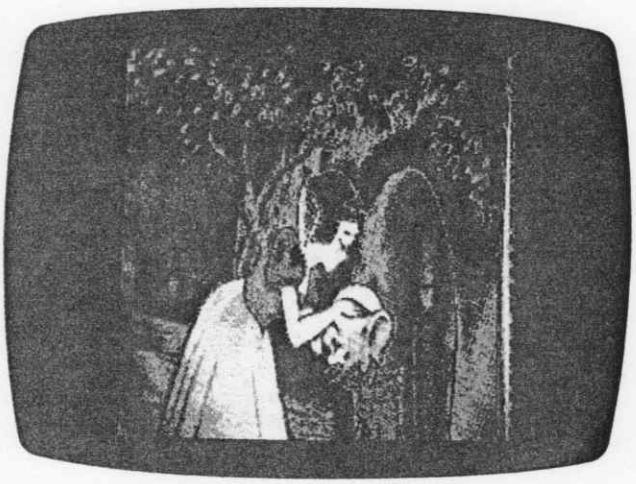


Photo 3: *Mapping gray scale to color-palette values, as in photo 2. This time, however, we select colors with more care.*

the dress will change color, too.

To correct this sort of color Ping-Pong, we assign palette position 0 to actually be the color gray or black. This improves the way the image looks and is the easiest way to overcome the effect. Since we blindly combined every four gray levels into each palette position, the only way to get a reasonable-looking picture, such as photo 3, is to use lots of gray and black.

While this often results in an acceptable final image, we need a more precise way of mapping the gray levels to the EGA palette. Instead of blindly mapping every group of four gray levels to each palette position, we can achieve better results if each gray level from the original image can be mapped to any of the 16 palette positions. This also serves to avoid the color Ping-Pong problem.

The easiest way to implement this mapping method is to start by displaying information for 16 gray levels at a time. We can do this by mapping gray levels 0 through 15 to palette positions 0 through 15. We assign any gray level greater than 15 to palette position 0. By making the color of palette position 0 black, this hides all the gray levels greater than 15.

The result of this interim mapping scheme is shown in photo 4. Notice that the pixels that make up Snow White and Dopey are not shown. They have disappeared because they are composed of gray levels higher than 15, which we have assigned to palette position 0 (black). Thus, an added side effect of this mapping technique is the ability to separate the image into layers of light levels.

Once we have the first 16 raw gray levels to work with, we start the task of assigning them to palette positions. The coloring software lets us assign any of the EGA's 64 colors to each of the 16 palette positions very rapidly. A quick selection of available colors gives us photo 5. (The colors I selected are not necessarily the ones used in *Snow White and the Seven Dwarfs* because the EGA palette has a limited range of colors. My objective is more the procedure rather than the actual color quality of the results.)

At this point in the process, we don't actually use 16 different colors but rather assign the 16 original gray levels individually to some limited number of colors. If you use all 16 colors for these first 16 levels, these will be the only 16 colors you can use when selecting palette positions later for the higher gray-level values. Try to use just four or five different colors that are spread throughout the palette.

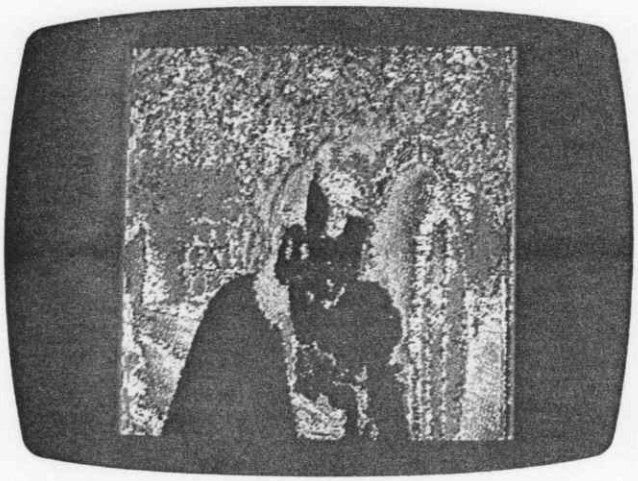


Photo 4: *Masking those gray levels mapped to palette positions greater than 15 simplifies the colorization process by letting you concentrate on 16 colors at a time, in this case, those pixels assigned to palette positions less than 16.*

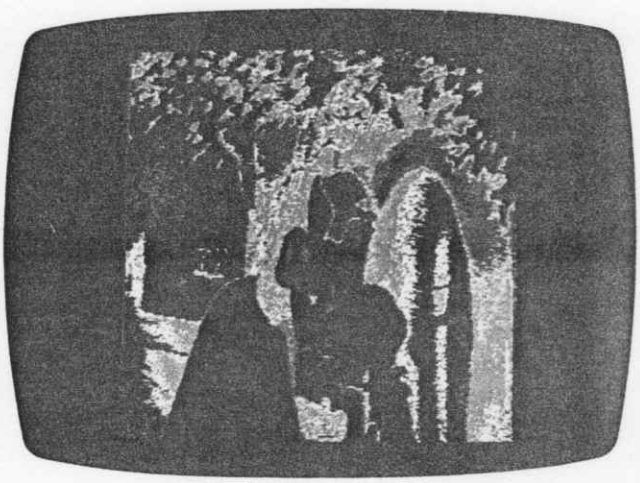


Photo 5: *Using the masked image from photo 4, you can rapidly assign colors to the displayed pixels.*

Photo 6: *A reverse mask to photo 4. Now you can colorize pixels mapped to palette positions greater than 15.*

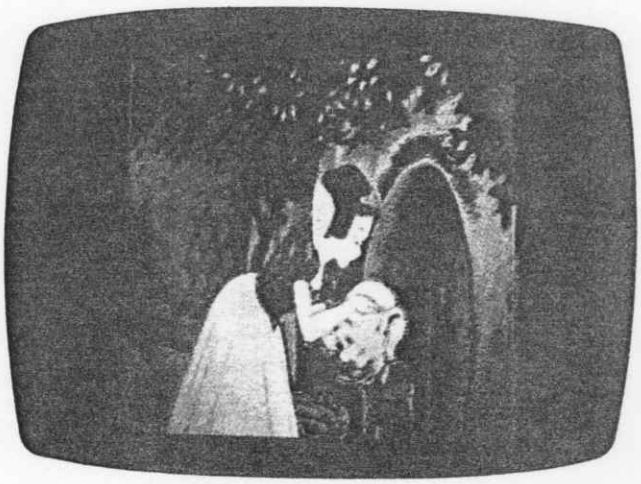


Photo 7: *Final result—the wonderful world of color.*

In photo 5, I used black, light green, dark green, light gray, dark gray, and brown (they become palette positions 0 through 5). The dark green color may be assigned to pixel gray-scale values of 4, 8, 13, and 14 as a typical example. Similarly, pixels with original gray-scale values of 0, 2, 3, and 15 might appear best if painted black. As you assign colors to a gray-scale value, you can see which pixels are affected and make a choice. Dark green leaves might look better against a blue sky than light green ones.

After we have selected the colors for the first 16 gray levels, we will repeat the process for the next 16 levels. We do this the same way except that this time we use gray-level values between 16 and 31. We will assign gray levels 0 through 15 and any levels greater than 31 to palette position 0 (black). As a result, gray levels 0 through 15 and those greater than 31 will not be visible.

Photo 6 shows levels 16 through 31 after we have selected colors. The new colors selected are purple, yellow, white, and pink. The software will combine these colors with the six we picked earlier, so the palette now has 10 colors assigned and 6 available for the remaining 32 gray levels. I repeated the color-selection process for levels 32 through 47 and levels 48 through 63, adding only a single new color, red.

As I mentioned earlier, grouping the gray-level data gave us a layered image even though that was not our intention. Snow White and Dopey are brighter than their surroundings and were digitized with higher gray-scale values. If these two fantasy characters were to be used independently or were more significant to you than the background colors, you could assign the majority of palette positions to their specific set of gray levels rather than the ascending order I described. Alternatively, we could make the background entirely black, dark gray, dark green, and brown to highlight Snow White. The effects that you can produce through the process of colorization are limitless (this is part of the controversy that is currently surrounding movie colorization).

After all 64 levels have been assigned a palette position, the coloring software will redraw the image, using the mapping and palette colors selected. The image you see in photo 7 shows the final results of our work after some minor reselection of colors.

Satisfactory Results

While the EGA's color capabilities are a limitation, careful manipulation of gray levels and color values can give us satisfactory results. One important fact to remember is that at no time have I rearranged, reprocessed, or otherwise manipulated the digitized pixel data in any way. We have achieved all of the results that you see here purely by assigning colors to specific gray-scale values from the original data.

One significant improvement to the coloring software would be the ability to load and save the final images in a format compatible with some of the PC drawing programs. You could use coloring software to map the gray levels and then the drawing program to fine-tune the images and borders between colors.

Some software of this type is already available. I have PC utility programs that convert ImageWise picture files to be compatible with PC paint programs such as PC Paintbrush, EGAPaint, and PC Paint Plus. Other utilities let you print a picture on a dot-matrix printer or a laser printer. (Contact CCI for details.)

For the past four months, I've tried to present the ImageWise system and software as the basis of a truly cost-effective image processing system. The response has been more than I could have hoped for, and ImageWise will join the Circuit Cellar all-time hit parade. Of course, that's until you see some of the other projects I've got up my sleeve.

Next Month

I'll show you the Circuit Cellar IBM PC AT–compatible computer, which consumes only 25 percent of the power of an AT, is 100 percent compatible, and is only the size of a PC expansion board. ■

Special thanks to Dave Lundberg for his software expertise and help on this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Book Company, P.O. Box 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

CCI
P.O. Box 428
Tolland, CT 06084
(203) 875-2751

1. ImageWise digitizer/transmitter board experimenter's kit. Contains digitizer/transmitter printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with transmitter software, and CA3306 flash A/D converter and manual with complete parts list.
DT01-EXP $99

2. ImageWise display/receiver board experimenter's kit. Contains gray-scale display/receiver printed circuit board, 11.05-MHz crystal, programmed 2764 EPROM with receiver software, Telmos 1852 video D/A converter, manual with complete parts list, and an IBM PC 2.0 disk containing sample digitized images and test patterns.
DR01-EXP $99
DT01-EXP and DR01-EXP together $179

3. ImageWise digitizer/transmitter full kit. Contains all digitizer/transmitter components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, CA3306 flash A/D converter, manual, and IBM PC 2.0 disk containing utility routines for storing and displaying (dot-dithered, not gray scale) and downloading image files using an IBM PC. Does not include power supply or case.
DT01-KIT $249

4. ImageWise display/receiver full kit. Contains all gray-scale display/receiver components, including printed circuit board, 64K bytes of static RAM, IC sockets, crystals, programmed 2764, Telmos 1852 video D/A converter, manual, and an IBM PC 2.0 disk containing sample digitized images and test patterns. Does not include case or power supply.
DR01-KIT $249
DT01-KIT and DR01-KIT together $489

ImageWise is also available assembled. Call CCI for source and availability of assembled boards and complete systems, black-and-white TV cameras, 32K-byte static RAM chips, and power supplies. Software utilities are also available in SB180 format.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Surface delivery (U.S. and Canada only): add $3 for U.S., $6 for Canada. For delivery to Europe via U.S. airmail, add $10. Three-day air freight delivery: add $8 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express). Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

To be included on the Circuit Cellar mailing list and receive periodic project updates and support materials, please circle 100 on the Reader Service inquiry card at the back of the magazine.